Week 12 In-Class Exercise

Clarification:

To work on this exercise:

- **Function __str__ is expected to be added to every class although it is not specified in the writing.**
- **Attributes should be declared as private unless stated otherwise.**
- **To access the value of each attribute, include getter (or accessor) method or set up property of function with the same as the attribute name accordingly.**
- **Limit the outside class NOT to change the value of each attribute unless it is necessary.**

Overall concept of this in-class exercise

- **This in-class exercise asks you to create a basic snake and ladder board game.**
- **To simplify the program, the game will be played by 2 players only.**
- **In each round, both players will toss a dice and move along the snake and ladder board game.**
- **In some cells of this board game, it will reward the player with more forward steps (equivalent to the ladder) or make player move backward for some steps (equivalent to the snake).**
- **For some cells in this board game, if the hold value of such cell is True, the player must stop playing for one round. No dice is tossed and no movement of this player occurs. This player allows to resume his/her play next round.**
- **Initially, before the game is played, let both players be in the first cell (with cell ID = 0).**
- **The board has the size = board_width (number of columns) x board_length (number of rows). There are 3 text files for 3 boards for you to test (board1.txt, board2.txt, board3.txt.)**
- **Inside each text file, there are 3 main parts: board width (first line), board length (second line), and information of board cells (remaining of text file). For board cell information, each line gives out 3 values: cell ID, number of moves to let player move when he/she at this cell, hold status (TRUE/FALSE).**
- **The last cell with maximum cell ID is considered as the winning cell. If one of players reaches this winning cell first, such player is considered as a winner.**
- **The game continuously is played until there is a winner.**

1. Define class Cell such that there are 4 attributes:
   - id: cell ID
   - move: number of moves that a player can move forward (positive value) or backward (negative value)
   - hold: hold status. If value of hold is True, then a player must stay still for one round.
   - occupy_list: list of Player objects that are currently at this cell. Set default value to be empty list. Note that we will define Player class later.

The partial code to run cell class is given below:

run_cell.py	Output

`from cell import *`
 	
`c1 = Cell(1,2,True)`
`print(c1)`
`print(c1.occupy_list)`
	
	`1,2,True`
	`[]`

2. Create another class called Board such that there are 3 attributes:
    - cell_list: list of Cells. Note that board is a list of cell objects.
    - length: length (number of rows) of board. Set default value to be zero.
    - width: width (number of columns) of board. Set default value to be zero.

    Note that values of 3 attributes will be obtained from the text file. Read information about the text file in the **overall concept of this in-class exercise (first page)**

    - Inside __init__, function read_file will be called to read board information from the text file.

Method	Functionality
read_board(filename:string)	Read information of board from file and use this information to construct a board

The partial code to run Board class is given below:

run_board.py	Output

`from board import *`
 	
`board = Board('board1.txt')`
`print(board.length)`
`print(board.width)`
`print(board)`
	4
	5
	0,0,False
	1,4,False
	2,2,False
	3,0,True
	4,0,False
	5,0,False
	6,-2,False
	7,0,False
	8,2,False
	9,0,True
	10,0,False
	11,-4,False
	12,0,True
	13,4,False
	14,0,False
	15,0,False
	16,-2,False
	17,0,False
	18,0,True
	19,0,False

3. Create another class called Player such that there are 4 attributes:
    - name: initial name of one player. Note that this name should be one single character since the name will be used to show when a board is printed.
    - current_pos: current position of a player. This position is equal to cell ID where the player is currently located. Set default value to be zero.
    - current_hold: hold status. If this value is True, the player must stop playing for one round. He/she can resume playing in the next round. If this value is False, the player allows to play as usual. Set default value to be False.
    - current_move: current move of a player. This value of move specifies how many moves this player is going to move next. Set default value to be zero.

The partial code to run Player class is given below:

run_player.py	Output

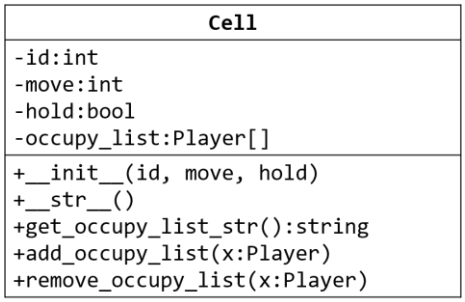
`from player import *`
 	
`a = Player('A')`
`b = Player('B')`
`print(a)`
`print(b)`
	`A: Pos = 0: Hold = False: Move = 0`
	`B: Pos = 0: Hold = False: Move = 0`

4. Next, we add 3 methods in class Cell: get_occupy_list_str, add_occupy_list, and remove_ occupy_list

   Information of these 3 methods are given in the table below:

Method	Functionality
get_occupy_list_str(): string	Create a string composed of names from all players in the occupy_list. Each name is separated by comma. Then, return this combined string. If there is no player in the occupy_list, return empty string. Example: if the occupy_list has two players with the names 'X', and 'Y', then this function will return a string of 'X,Y,' as the output of function.
add_occupy_list(x:Player)	Add player x to the occupy_list
remove_occupy_list(x:Player)	Remove Player object in the occupy_list with the same name as player x. Assume that no two players have the same name.

Note that the class diagram of class Cell can be shown as followed:



The code for run_cell and its output are given below:

run_cell.py

```
from cell import *
from player import *

c1 = Cell(1,2,True)
print(c1)

a = Player('A')
b = Player('B')
print(a)
print(b)

# Testing get_occupy_list_str and add_occupy_list
print(c1.get_occupy_list_str()) # no player in
the occupy_list, so this line prints empty string
c1.add_occupy_list(a)
print(c1.get_occupy_list_str())
c1.add_occupy_list(b)
print(c1.get_occupy_list_str())

# Testing remove_occupy_list
c1.remove_occupy_list(a)
print(c1.get_occupy_list_str())
c1.remove_occupy_list(b)
print(c1.get_occupy_list_str())  # no more player
in the occupy_list, so this line prints empty
string
```

Sample Output

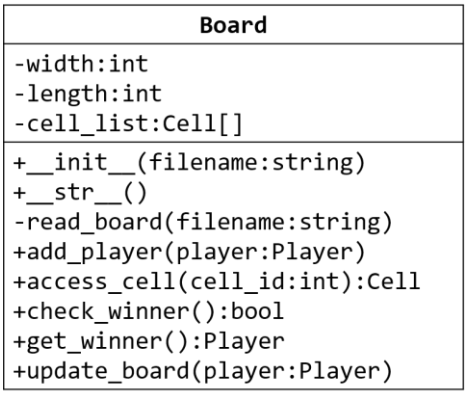
```
1,2,True

A: Pos = 0: Hold = False: Move = 0
B: Pos = 0: Hold = False: Move = 0

A,

A,B,

B,
```

5. Next, we will add 4 methods inside class Board: add_player, access_cell, check_winner, and get_winner.

   Information of these 4 methods are given in the table below (along with read_board function that we work on in question 2 and another function update_board that we will work on in question 7) :

Method	Functionality
read_board(filename:string)	Read information of board from the text file with filename and use this information to construct a board.
add_player(player:Player)	Add player to the board. This is equivalent to add the player to the first cell (with cell_id = 0). Note that function add_player will be called before the game is played.
access_cell(cell_id:int):Cell	Return the cell with the specified cell_id
check_winner():bool	Return True if there is winner. Otherwise, return False if the game is still playing and there is no winner.
get_winner():Player	If there exists winner of the game, return the player who is the winner. Otherwise, return None.
update_board(player:Player)	Update board with the information of specified player

Note that the class diagram of class Board can be shown as followed:



The code for run_board2.py to test add_player, access_cell, check_winner, get_winner functions and its output are given below:

run_board2.py	Sample Output

`from player import *`
`from board import *`
`from cell import *`
 	
`a = Player('A')`
`b = Player('B')`
`print(a)`
`print(b)`
 	
`board = Board('board1.txt')`
 	
`print('>>> Test function add_player:')`
`board.add_player(a)`
`print(board.cell_list[0].get_occupy_list_str())`
`board.add_player(b)`
`print(board.cell_list[0].get_occupy_list_str())`
 	
`print('>>> Test function access_cell:')`
`starting_cell_id = 0`
`starting_cell = board.access_cell(starting_cell_id)`
	`A: Pos = 0: Hold = False: Move = 0`
	`B: Pos = 0: Hold = False: Move = 0`
	
	`>>> Test function add_player:`
	
	`A,`
	
	`A,B,`
	
	`>>> Test function access_cell:`

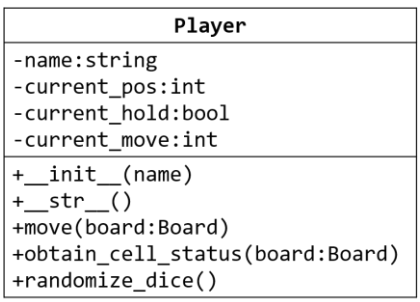
```
print(starting_cell)
print(starting_cell.get_occupy_list_str())

winning_cell_id = ____?_____  # Fill ? yourself
winning_cell = board.access_cell(winning_cell_id)
print(winning_cell)
print(winning_cell.get_occupy_list_str()) # print empty
string since no player in the winning cell

print('>>> Test functions check_winner & get_winner when
there is no winner:')
print(board.check_winner())
print(board.get_winner())

print('>>> Add player a to winning cell')
winning_cell.add_occupy_list(a)
print(winning_cell)
print(winning_cell.get_occupy_list_str())

print('>>> Test functions check_winner & get_winner when
there is a winner:')
print(board.check_winner())
print(board.get_winner())
```

```
0,0,FALSE
A,B,



19,0,FALSE



>>> Test functions check_winner &
get_winner when there is no winner:
False
None

>>> Add player a to winning cell

19,0,FALSE
A,

>>> Test functions check_winner &
get_winner when there is a winner:
True
A: Pos = 0: Hold = False: Move = 0
```

6. Next, we will add 3 methods inside class Player: move, obtain_cell_status, randomize_dice. Information of these 3 methods are given in the table below:

Method	Functionality
move(board:Board)	Update current_pos value of player to the corresponding cell, based on the current_move value of player. After moving, set the current_move value of player to zero. If the corresponding cell is more than what is inside the board, move player to the last cell (which is the winning cell.)
obtain_cell_status(board:Board)	Obtain the status of the current cell where the player is located at. Assign *the status of the current cell* as the status of player. Equivalently, update current_hold and current_move of player with the status of current cell. Note that obtain_cell_status function ONLY obtain cell status but DO NOT move player to the new cell.
randomize_dice()	Randomize dice point (between 1-6 , inclusive) for the player and assign this random value to current_move value of player.

Note that the class diagram of class Player can be shown as followed:



The code for run_board3.py to test move, obtain_cell_status, randomize_dice functions and its output are given below:

run_board3.py	Sample Output

`from player import *`
`from board import *`
`from cell import *`
 	
`a = Player('A')`
`b = Player('B')`
 	
`board = Board('board1.txt')`
`board.add_player(a)`
`board.add_player(b)`
 	
`###  For testing function move`
`print('>>> A moves')`
`print(a)`
`# Set player A to move 4 steps`
`a.current_move = 4`
`print(a)`
`a.move(board)`
`print(a)`
 	
`print()`
`print('>>> B moves 1 step and obtains new cell status')`
`print(b)`
`# Set player B to move 1 step`
`b.current_move = 1`
`print(b)`
`b.move(board)`
`print(b)`
`print('>>> Print B\'s cell 1 status.')`
`cell = board.access_cell(b.current_pos)`
`print(cell)`
`###  For testing function obtain_cell_status`
`print('>>> B obtains current cell 1 status')`
`b.obtain_cell_status(board)`
`print(b)`
`# B is still at cell 1.  Function obtain cell status does not move B.`
`# To update move=4 obtained from cell 1, we need to call function move again.`
`print('>>> After obtaining cell 1 status and moving B:')`
`b.move(board)`
`print(b)`
 	
`print()`
`print('>>> A moves 5 steps and obtains new cell status')`
`print(a)`
`# Set player A to move 5 steps`
`a.current_move = 5`
`print(a)`
`a.move(board)`
`print(a)`
`a.obtain_cell_status(board)`
`print(a)`
`print('>>> Print current A\'s cell')`
`cell = board.access_cell(a.current_pos)`
`print(cell)`
 	
`###  For testing function randomize_dice`
`print()`
	`>>> A moves`
	`A: Pos = 0: Hold = False: Move = 0`
	
	`A: Pos = 0: Hold = False: Move = 4`
	
	`A: Pos = 4: Hold = False: Move = 0`
	
	`>>> B moves 1 step and obtains new cell status`
	`B: Pos = 0: Hold = False: Move = 0`
	
	`B: Pos = 0: Hold = False: Move = 1`
	
	`B: Pos = 1: Hold = False: Move = 0`
	`>>> Print B's cell 1 status.`
	
	`1,4,FALSE`
	
	`>>> B obtains current cell 1 status`
	
	`B: Pos = 1: Hold = FALSE: Move = 4`
	
	`>>> After obtaining cell 1 status and moving B:`
	
	`B: Pos = 5: Hold = FALSE: Move = 0`
	
	`>>> A moves 5 steps and obtains new cell status`
	
	`A: Pos = 4: Hold = False: Move = 0`
	
	`A: Pos = 4: Hold = False: Move = 5`
	
	`A: Pos = 9: Hold = False: Move = 0`
	
	`A: Pos = 9: Hold = TRUE: Move = 0`
	`>>> Print current A's cell`
	
	`9,0,TRUE`

```
print('>>> B randomizes dice, moves, and obtain new
cell status')
print(b)
b.randomize_dice()   #  randomizes dice.  Your
randomized value may not be 6, like the sample output.
print(b)
b.move(board)   # move
print(b)
b.obtain_cell_status(board)  # obtain new cell status
print(b)
print('>>> Print current B\'s cell')
cell = board.access_cell(b.current_pos)
print(cell)

print('>>> B moves more than what is inside the board.
       B will move to the last cell.')
b.current_move = 30
print(b)
b.move(board)
print(b)

winning_cell_id = __?___  # Fill ? yourself
winning_cell = board.access_cell(winning_cell_id)
print(len(winning_cell.get_occupy_list_str()))
# The result is zero although b is at the winning cell.
# This happens because function move does not add the
player t0 occupy_list of the cell. We must explicitly
update occupy_list after running function move.

print(board.check_winner())
print(board.get_winner())
```

```
>>> B randomizes dice, moves, and obtain new
cell status
B: Pos = 5: Hold = FALSE: Move = 0


B: Pos = 5: Hold = FALSE: Move = 6

B: Pos = 11: Hold = FALSE: Move = 0

B: Pos = 11: Hold = FALSE: Move = -4
>>> Print current B's cell

11,-4,FALSE

>>> B moves more than what is inside the board.
B will move to the last cell.

B: Pos = 11: Hold = FALSE: Move = 30

B: Pos = 19: Hold = FALSE: Move = 0



0









False
None
```

7. In class Board, add function update_board with information of the specified player.

   A function update_board is called after player randomizes dice point. Thus, inside function update_board will:

   a) Move the player to new cell,
   b) Obtain status of new cell and assign to player's current_hold and current_move
   c) Update the occupy_lists of the old cell and the new cell of player

   Note that steps a),b),c) will be repeated over and over until the player moves to the cell with the move value equal to zero.

   The code for run_board4.py to test function is given below:

run_board4.py

```
from board import *
from player import *

board = Board('board1.txt')

a = Player('A')
b = Player('B')
board.add_player(a)
board.add_player(b)


print('Starting...')
print(board.cell_list[0].get_occupy_list_str())

round = 1
print('>>> Round ' + str(round))
```

```
player = a
player.randomize_dice()
print(player.name + '\'s position = ' + str(player.current_pos) + '. ',end='')
print(player.name + ' moves ' + str(player.current_move) + ' steps.')

board.update_board(player)
print(player)
print(board.cell_list[a.current_pos])
print(board.cell_list[a.current_pos].get_occupy_list_str())
print(board.cell_list[0].get_occupy_list_str())
```

The sample outputs are given below

Sample Output 1	Sample Output 2
Starting... A,B, >>> Round 1 A's position = 0. A moves 5 steps. A: Pos = 5: Hold = FALSE: Move = 0 5,0,FALSE A, B,	Starting... A,B, >>> Round 1 A's position = 0. A moves 3 steps. A: Pos = 3: Hold = TRUE: Move = 0 3,0,TRUE A, B,
Sample Output3	Sample Output 4
Starting... A,B, >>> Round 1 A's position = 0. A moves 2 steps. A: Pos = 4: Hold = FALSE: Move = 0 4,0,FALSE A, B,	Starting... A,B, >>> Round 1 A's position = 0. A moves 6 steps. A: Pos = 4: Hold = FALSE: Move = 0 4,0,FALSE A, B,

8. From question 7, add the followings to have the complete game:
   - In class Board, function update_board, add the code to handle when current_hold value of player is True.
   - In run_board, rewrite the code, so that the game will continue to play until one of players reach the last cell (the winning cell) first.
   - In class Board, rewrite __str__ function, so it will print board as 2D board game as shown.
   - Randomly select one of player to play first. (For example, in sample output 1 below, A plays first. In sample output 2, B plays first.)

Sample Output 1	Sample Output 2
Starting... ----------------------------- | 0 | 1 | 2 | 3 | 4 | | 0,F | 4,F | 2,F | 0,T | 0,F | |A,B, | | | | | ----------------------------- | 5 | 6 | 7 | 8 | 9 | | 0,F |-2,F | 0,F | 2,F | 0,T | | | | | | | ----------------------------- | 10 | 11 | 12 | 13 | 14 | | 0,F |-4,F | 0,T | 4,F | 0,F | | | | | | | ----------------------------- | 15 | 16 | 17 | 18 | 19 | | 0,F |-2,F | 0,F | 0,T | 0,F | | | | | | | -----------------------------	Starting... ----------------------------- | 0 | 1 | 2 | 3 | 4 | | 0,F | 4,F | 2,F | 0,T | 0,F | |B,A, | | | | | ----------------------------- | 5 | 6 | 7 | 8 | 9 | | 0,F |-2,F | 0,F | 2,F | 0,T | | | | | | | ----------------------------- | 10 | 11 | 12 | 13 | 14 | | 0,F |-4,F | 0,T | 4,F | 0,F | | | | | | | ----------------------------- | 15 | 16 | 17 | 18 | 19 | | 0,F |-2,F | 0,F | 0,T | 0,F | | | | | | | -----------------------------

```
>>> Round 1
A's position = 0. A moves 4 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
| B,  |     |     |     | A,  |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

B's position = 0. B moves 2 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |A,B, |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

>>> Round 2
A's position = 4. A moves 2 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |B,A, |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

B's position = 4. B moves 3 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     | A,  |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
```

```
>>> Round 1
B's position = 0. B moves 6 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
| A,  |     |     |     | B,  |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

A's position = 0. A moves 5 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     | B,  |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
| A,  |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

>>> Round 2
B's position = 4. B moves 5 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
| A,  |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

A's position = 5. A moves 5 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
```

```
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     | B,  |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

>>> Round 3
A's position = 4. A moves 4 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     | B,  |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
| A,  |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

B's position = 7. B moves 4 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     | B,  |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
| A,  |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------


>>> Round 4
A's position = 10. A moves 3 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     | B,  |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
```

```
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
| A,  |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

>>> Round 3
B holds.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
| A,  |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------

A's position = 10. A moves 4 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     | A,  |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------


>>> Round 4
B's position = 9. B moves 6 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     | A,  |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
```

```
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     | A,  |     |     |
-------------------------------

B's position = 7. B moves 2 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     | A,  |     |     |
-------------------------------

>>> Round 5
A's position = 17. A moves 4 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     | B,  |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     |     |     | A,  |
-------------------------------

Game over. A wins!
```

```
| 0,F |-2,F | 0,F | 0,T | 0,F |
| B,  |     |     |     |     |
-------------------------------

A's position = 14. A moves 1 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|B,A, |     |     |     |     |
-------------------------------

>>> Round 5
B's position = 15. B moves 2 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
| A,  |     | B,  |     |     |
-------------------------------

A's position = 15. A moves 5 steps.
-------------------------------
|  0  |  1  |  2  |  3  |  4  |
| 0,F | 4,F | 2,F | 0,T | 0,F |
|     |     |     |     |     |
-------------------------------
|  5  |  6  |  7  |  8  |  9  |
| 0,F |-2,F | 0,F | 2,F | 0,T |
|     |     |     |     |     |
-------------------------------
| 10  | 11  | 12  | 13  | 14  |
| 0,F |-4,F | 0,T | 4,F | 0,F |
|     |     |     |     |     |
-------------------------------
| 15  | 16  | 17  | 18  | 19  |
| 0,F |-2,F | 0,F | 0,T | 0,F |
|     |     | B,  |     | A,  |
-------------------------------

Game over. A wins!
```

OPTIONAL: You can edit this game further such as

- Allow more than 2 players can play this game. You may need to adjust the __str__ function to adapt printing out board.
- Allow player to hold more than one round. Instead of True/False value, you can change it to integer value to specify how many rounds the player must hold.
- Redraw printing 2D board game such that the cells in the lower row actually continue from the cells in the upper row. See example below.

```
-------------------------------
|  0   |  1   |  2   |  3   |  4   |
| 0,F  | 4,F  | 2,F  | 0,T  | 0,F  |
|A,B,  |      |      |      |      |
-------------------------------
|  9   |  8   |  7   |  6   |  5   |
| 0,T  | 2,F  | 0,F  |-2,F  | 0,F  |
|      |      |      |      |      |
-------------------------------
| 10   | 11   | 12   | 13   | 14   |
| 0,F  |-4,F  | 0,T  | 4,F  | 0,F  |
|      |      |      |      |      |
-------------------------------
| 19   | 18   | 17   | 17   | 15   |
| 0,F  | 0,T  | 0,F  |-2,F  | 0,F  |
|      |      |      |      |      |
-------------------------------
```