README

Ball Sort Game

Video: https://youtu.be/OFOEqBTxC3g

Intro

Ball Sort Game is self-explanatory, the user aims to sort the balls in each tube by color.
Rules:

1) The user can only move the top ball of each tube to another tube (non-empty tube)
2) The user cannot move to and from the same tube
3) The user can only move to tubes that are not full
4) The game ends when all colors are sorted in each tube

Project Approval

TA approved the project, and minor feature changes were made, including using shuffled lists to represent randomized ball positions, and text UI to visualize the game and movement of balls across tubes.

Instructions

1. User selects ENTER to move forward in each step
2. User selects the source tube by entering an integer (0-2) indicating the tube
3. User then selects the destination tube by entering an integer (0-2) indicating the tube

Features

1. Randomization of ball positions - shuffled lists (balls in tubes)
2. Text UI - visualization of adding and removing elements from the list indicates the movement of balls among tubes (shown in the terminal)

Justification for Complexity

- List objects with elements to represent each ball in the tube
- Print the list of objects vertically to visually represent the balls in each tube and the movement of balls across tubes
- The user can only move the top ball of each tube to another tube (removing the lowest-index item of each tube and instantly updating each tube)
    - Implementation (keyboard control):
        - ENTER: move forward & continue through each step
        - Enter integer 0-2: select the source and destination tube
- Levels vary every time: lists are shuffled and there is randomization of ball placement

Lists & Script Variables

- Variables
    - sourceTube - input variable: choose from which tube
    - destTube - input variable: destination tube of ball
- List
    - new_board: copy of original board (to update positions within function each time)
    - Board_lst: 12 items in total - 4 “R”s, 4 “B”s, 4 “ ” (empty strings)
    - Board_shuffled: shuffled Board_lst and subset the list into 3 portions (4 items per list), rearrange each sublist by counting the empty spaces and putting the empty spaces at the beginning of the list
        - I.e. Board_lst: ["R", "B", "R", "B", "B", "R", "B", "R", " ", " ", " ", " "]
        - Board_shuffled (sublist -> rearranged):
            - shuffled(Board_lst): ["R", " ", " ", "B", "R", "R", "B", "R", " ", "B", "B", " "]
            - ["R", " ", " ", "B"] -> [" ", " ", "R", "B"]
            - ["R", "R", "B", "R"] -> ["R", "R", "B", "R"]
            - [" ", "B", "B", " "] -> [" ", " ", "B", "B"]

Function Table

Block / Function Name	**Domain (inputs)**	**Range (outputs)**	**Behavior (role in the context of the project)**
Game functions			
__main__()	empty	empty	Run the game automatically
game()	empty	empty	Run the features of the game 1. Print instruction statements 2. Ask user for inputs (sourceTube and destTube integer)

random_board()	list	list	Take in a list: - count the number of spaces in the list; - put the items together in the list such that there are no empty spaces in between or spaces that come after the items (representing the scenario where balls are stuck at the bottom of each tube) - append the number of spaces back to the beginning of each respective tube
check_valid_move()	1. Board (nested list) 2. sourceTube (int) 3. destTube (int)	Boolean (TRUE=valid move)	Can ONLY move if: 1. destTube has at least one empty space 2. sourceTube is not empty 3. sourceTube != destTube
update_board()	1. Board (nested list) 2. sourceTube (int) 3. destTube (int)	Board (nested list)	Update the position of the board if the move is valid: 1. Check empty spaces for both sourceTube and destTube 2. Identify the last ball of sourceTube and identify the first space of destTube a. choose ball from sourceTu

			be and place the ball in the first space of destTube
solved()	Board (nested list)	Boolean (TRUE=game solved)	Check if the conditions for the game being solved are met: 1. All items in any tube == “R” 2. And all items in any tube == “B”
Test functions			
test_update_board(self)	1. Boards (nested lists) 2. sourceTube (int) 3. destTube (int)	True (updated)	Checks to see if board1 will update into board2
test_solvedboard_solved(self)	Board (nested list)	True (solved)	All balls are sorted (all items in any tube is the same element) is considered solved board
test_solvedboard_notsolved(self)	Board (nested list)	False (not solved)	Any tube has mixed elements is considered not solved
test_check_valid_move_valid(self)	1. Board (nested list) 2. sourceTube (int) 3. destTube (int)	True (valid)	A move is considered valid if the source tube is not empty, and the destination tube has at least one empty space, and the source tube is different from the destination tube.
test_check_valid_move_invalid(self)	1. Board (nested list) 2. sourceTube (int) 3. destTube (int)	False (invalid)	A move is considered invalid if the source tube is empty, or the destination tube is full, or source tube is the same as destination tube.