Name: Harshul Gupta
College Name: ABES Engineering College
Email ID: harshul.18bci1055@abes.ac.in
Problem Domain: Front-End and Dashboard Developer
Problem Statement: 2048-Game
GitHub Link - https://github.com/Harshul555/2048-Game.git

Design Principles:

- The code has been made using the following python and its following packages:
  - Tkinter: for making the game gui
  - Numpy: for making quick mathematical calculations
  - Random: for randomly choosing the position within the grid for either 2 or 4
- The code has been divided into two classes: Board and Game

Problems Faced during designing the solution:

- The difficult task was how to perform the required changes in the grid whenever the player made any move
- Whenever the player makes any move, all the filled blocks are shifted in that direction
- The blocks with the same value needed to be combined
- For performing such actions for all the moves different operations are required
- For the moves like above and down, the matrix needs to be transposed
- For the moves like right the matrix needs to be reversed along the rows to make calculation easier
- For the left move only compress and merging operations are sufficient
- For performing such moves, one need to come up with the valid combination of moves
- The player might want to play the game again, for that a button needs to be provided at the end

Solution:

- For making the required changes a combination of different moves is made for each move the player chose to perform.

1. Up:
   transpose
   compress
   merge
   check if a valid move was made
   compress
   transpose

2. Down:
   transpose
   reverse

compress
merge
check if a valid move was made
compress
reverse
transpose

3. Left:
   compress
   merge
   check if a valid move was made
   compress

4. Right:
   reverse
   compress
   merge
   check if a valid move was made
   compress
   reverse

- A confirmation message is shown to the user to know if they want to play another game or not.

Code Walkthrough:

- Within the class Board:
  - The bg_color – a dictionary for setting up the back ground colors of blocks of different numbers to make the game more interactive
  - The init method - the constructor of the Board class has a default argument n having value 4. This constructor creates the Tkinter window, adds the title, creates the canvas within which the grid will be formed and the gridCell array for storing the array which hold the values which will be displayed on the game screen
  - A for loop which will be used to design the way each individual cell will be displayed
  - The reverse function- this function uses np.flip method to reverse the gridCell array along the rows
  - The transpose function – this function makes use of the np.transpose method which is used to interchange the rows and columns to make the calculation easy for moves like up and down
  - The compress function – this function uses a for loop and a np.pad function, this function to shift all the non-zero values into the left direction. It first removes all non-zero elements and then makes use of the np.pad method to add required zeros at the right side of each row
  - The merge function – this function is used to merger the blocks containing the same value into one block with twice the value, i.e., two blocks of 8 merge to become a single block of 16.

- The random_cell function - this function finds the index of all the blocks with empty or zero values and stores the index into a list. Then a random index is chosen using the random.choice method and a random value of either 2 or 4 is assigned to it
- The can_merge function – this function is used to check if there are any two adjacent blocks present in the game board which holds the same value and can be merged to create a new single block
- The paintGrid method – this method is used to paint or redesign the blocks of the game board after each successful moves made by the player

- Within the class Game:
  - The init method - the constructor is used to assign the board to the object attribute board, sets the end, won and st as false which helps determine whether the game has ended, the player has won or the player wants to play again respectively.
  - The start method – this method calls the random_cell method twice to present two initial blocks when the game starts, then calls the paintGrid method to create the canvas of the game area, and also uses the window.bind method to capture the key strokes and what action to take in response to them. The window.mainloop method opens up the game window
  - The check_win method: it is used to check if the player has achieved the 2048 target
  - The link_keys method – it takes the event (the key strokes by the player) as parameters, checks if the game has end or the player has won. Sets the attributes initially to false which will be used to know whether the merger, compress and move operations occurred successfully.
    Then the if else ladder has been used to check which move was made and the corresponding action has been performed. For all the moves the the combination of all the moves need to be performed in the following manner:

    1. Up:
       transpose
       compress
       merge
       check if a valid move was made
       compress
       transpose

    2. Down:
       transpose
       reverse
       compress
       merge
       check if a valid move was made
       compress
       reverse
       transpose

    3. Left:
       compress

merge
check if a valid move was made
compress

4. Right:
   reverse
   compress
   merge
   check if a valid move was made
   compress
   reverse

- Within the link_key method after making the changes as per the move made by the player, the paint_grid method is called to redesign the grid
- After that check_win methods is called. Also, it is checked if the game board has any space left to make a new move or if the compression can be done or not
- On the basis of these verifications, the result is shown to the user in the form of a message box along with the score they achieved
- Then a message box is used to ask if the player wants to continue or not, if yes the st is assigned True and returned.

- The main function:
  - An object of the class Board is made
  - That object of class Board is passed a parameter for the object of class Game
  - The Game.start method is then called to begin the game
  - The st value is checked, if it is True then the main function is called again, else the game terminates

Scope of making this an 8x8 from 4x4:

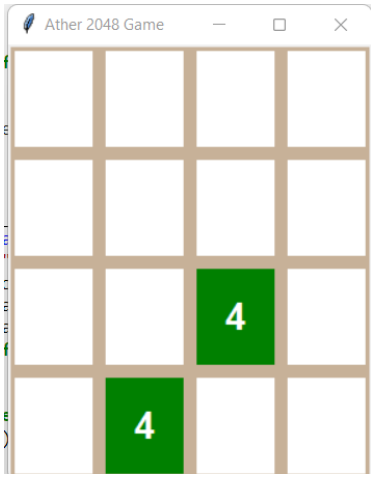
- The n value used to determine the size of the grid can be changes and the game for a 8x8 matrix can be made easily

Change from 2048 to 4096 as end number:

- In the check_win method of the Game class, the loop which checks the winning condition can be changed to have the winning value as the 4096 easily.

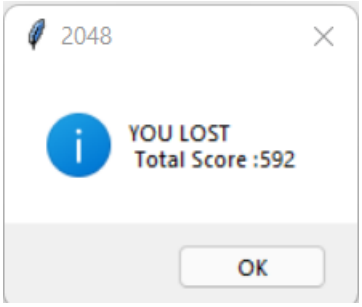
Outputs:

- Screen when the game begins-



- Fully populated game board-

- Message box when the use lose-



- Message box to confirm if the player wants to play again –