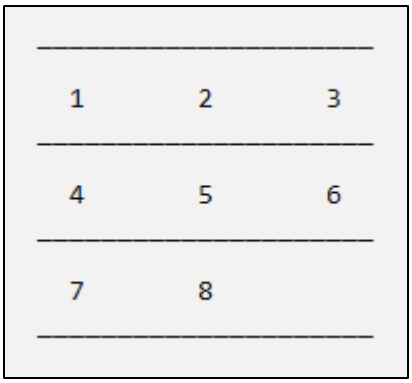
Number Puzzle Game

Objective

- Slide the numbers around until they are in numerical order from least to greatest, left to right and top to bottom.
- The white space is the open space that the numbered squares are moved to.
- To move a number present above and below the open space, press down arrow and up arrow, respectively.
- Similarly, to move a number that is present to the left and to the right of the open space, press right arrow and left arrow, respectively
- Only numbers that are immediately to the left, right, above or below the empty space can be moved.
- The open space should end up in the lower right corner when the puzzle has been solved.



Code Explanation:

Main Code:

A) First, the screen is cleared and the objective of the game is displayed

B) Then, the available m x m puzzles are listed, from which the player should select one

C) Depending upon the player's choice of 'm', an ideal version of the solved matrix is generated for reference and the position of the empty space ('pos' variable) is initialized to m*m (bottom right corner)

D) *checklist* and *check-dict* respectively contain the valid key press inputs and the displacement between the vacant space's next position (when player presses a key present in checklist) from its current position

E) The puzzle matrix 'r' is created using the create_mat(m) function and displayed using print_mat() function

F) Every time a key is pressed, we check whether it is a valid keypress belonging to *checklist* [Up, Right, Down, Left]. If yes, we determine the displacement between the present and new positions of empty space and see whether the new location is valid (For example: moving left from position 4 to position 3 and moving right from last position are invalid in a 3 x 3 puzzle)

G) If it is a valid move, then the new position(*new_pos* variable) is calculated inside *find_newpos()* function which adds the displacement to *pos*

H) Then the puzzle 'r' is modified (actual change to empty space's location) by calling the modify() function, and the pos is changed to the new position of the vacant space

I) The altered puzzle is printed, after clearing the screen, and checked whether it has been solved, by calling the *check()* function

J) If it has been solved, the player is congratulated and asked whether he wants to play again. His choice will decide whether the game will start from the beginning once again or will exit

K) If the puzzle has not yet been solved by the player, the game continues and waits for him to make the next move

Functions:

solvable (temp, ideal):

This function receives the 1D arrays *temp* and *ideal* and keeps solving the *temp* array(by using the method mentioned in: *https://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/*) until it becomes equal to *ideal* array. We count the number of moves that was needed to solve the puzzle using the method of inversion. If the count is even, that means the puzzle is of even parity and so it can be solved using normal method. So we return True; else if the count turns out to be odd, we return False

create_mat (m):

This function creates the m x m puzzle matrix 'r' and fills it with random values from 1 to (m x m) -1. A 1-D array, *temp,* is created with the same values as the puzzle matrix 'r'. Another array, *ideal,* which is nothing but the sorted version of *temp,* is also created. To check whether the puzzle is solvable (Puzzle with even parity), *solvable* function is called by passing *temp* and the *ideal* as parameters. If it is not solvable (odd parity Puzzle), an extra parity is added by swapping the first two elements of 'r' matrix to make it solvable (the parity is increased by one => even parity => solvable puzzle)

pos_verify ():

Pos_*verify* checks whether the new position of the empty space, determined using *find_newpos(),* is valid or not. If the new position is beyond the scope of the puzzle (say, new_pos = 12, in 3x3 puzzle) or results in an invalid move (for eg: new_pos=4 when pos=3), the function returns False. Else, the return value is True.

print_mat (m):

This function prints the m x m puzzle 'r' on the screen inside a box like appearance created using underscores. This function is called every time there's a change in the puzzle

modify (n):

The the puzzle is altered every time with the help of this function. It just swaps the empty space with the element which the user wanted to move (into it). The value of the current position of the empty space, *pos,* is altered before the function ends.

find_newpos ():

It finds the new position (using the value from *check_dict*) corresponding to the key pressed, considering the current location of empty space. For example, in a 3x3 puzzle, if the current position of the empty space is 5 and the pressed key is 'down arrow', then the *new_pos* is *new_pos=pos+m.* That is, *new_pos*=5+3 = 8.

check ():

check() function checks whether the puzzle has been solved by checking whether all the elements, with empty space at the last, are sorted in ascending order. If yes, it returns True, else returns False. If the return value is false, the while loop in the code repeats again and asks for player to make the next move.

The Game:

```
PuZzLe GaMe

Objective:

    $ Slide the numbers around until they are in numerical order from least to greatest, left to right and top to bottom.
    $ The white space is the open space that the numbered squares are moved to.
    $ To move a number present above and below the open space, press down arrow and up arrow, respectively.
    $ Similarly, to move a number that is present to the left and to the right of the open space, press right arrow and left arrow, respectively.
    $ Only numbers that are immediately to the left, right, above or below the empty space can be moved.
    $ The open space should end up in the lower right hand corner when the puzzle has been solved.

    Enjoy!



      _____________________
       1      2      3
      _____________________
       4      5      6
      _____________________
       7      8
      _____________________

Hit 'enter' key to begin
```

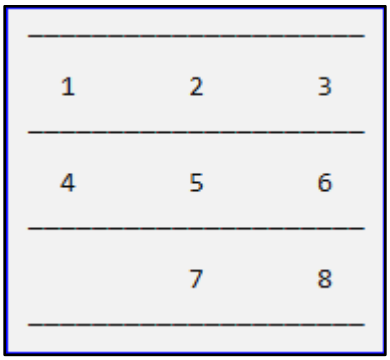
```
Select the dimensions of the puzzle
1) 3x3
2) 4x4
3) 5x5
4) 6x6
5) 7x7
6) 8x8

 Enter Your Choice: 1
```

```
 _____ _____ _____
|     |     |     |
|  7  |  1  |  2  |
|_____|_____|_____|
|     |     |     |
|  6  |  3  |  8  |
|_____|_____|_____|
|     |     |     |
|  5  |  4  |     |
|_____|_____|_____|
```

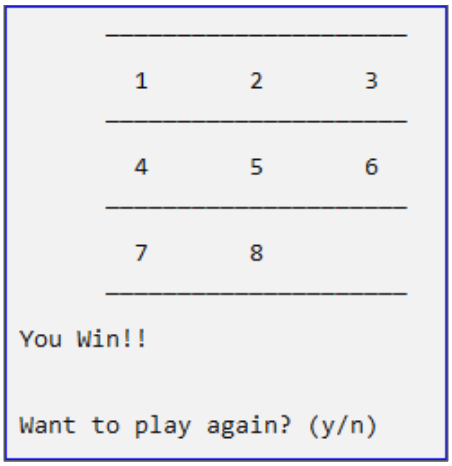
The Puzzle

```
 _____ _____ _____
|     |     |     |
|  1  |  2  |  3  |
|_____|_____|_____|
|     |     |     |
|  8  |  7  |  4  |
|_____|_____|_____|
|     |     |     |
|  6  |  5  |     |
|_____|_____|_____|
```

Solving the first row



Solving the second row



Solving the third row