Chapter 6: 2048

Description: 2048 uses direction keys to slide blocks into themselves, the same number of blocks will double, trying to get the highest score possible.
Resources:

- Python 3

Why?:

- Recursive programming is the step on a function going through every possibility dynamically. Very useful to know and remember.

To-do:

- Random put in number 2
- Slide-in different directions and collapse
- Extra random, any number can appear
- Extra hard make it (1 instead of 2 that appears)
- Multiple numbers are added each turn

Code:

Step 1:

```
import random

score = 0
gameOver = False

def instructions():
   print("The number 2 will appear in random spots and you will need to combine
numbers to save space")
   print("If you run out of room the game ends")
   print("Commands are as follows : ")
   print("Up key or 'w' : Move Up")
   print("Down key or 's' : Move Down")
   print("Left key or 'a' : Move Left")
   print("Right key or 'd' : Move Right")
   print()

def main_game(n):
   global gameOver
   board = create_board(n)

    while not gameOver:
```

```
        add_2(board, n)
        print_board(board,n)
          # move up, down, right, left
          # print

def print_board(board,n):
    global score
    for i in range(n):
        for j in range(n):
            val = board[i][j]
            if val > score:
                score = val
            print(f"|{val}|", end = '')
        print()
    print()

def create_board(n):
    board = []
    for i in range(n):
        board.append([0]*n)
    return board

# if all spots on board is taken
# if all values have no same values above or below
# regression
def game_over():
    global score, gameOver
    gameOver = True
    print(score)

# add a 2 to a random spot with a 0
def add_2(board, n):
    global gameOver
    inner_count = 0

    zero_x =[]
    zero_y =[]

    for i in range(n):
        for j in range(n):
            if(board[i][j] != 0):
                inner_count+=1
```

```
            else:
                zero_x.append(i)
                zero_y.append(j)

    if(inner_count == n*n):
        game_over()

    if(gameOver):
        return

    pos = random.randint(0,len(zero_x)-1)
    board[zero_x[pos]][zero_y[pos]] = 2
if __name__ == '__main__':
    instructions()
    main_game(4)
```

* This might be the first time you see a 2d array here is a visual
[
        [0,0,0]
        [0,0,0]
        [0,0,0]
]

How do we go through this? X and y coordinates
Board[0][0] would be the top left corner and then board[0][1] will be top middle and so on

Import random

Score is set to 0
gameOver is false

Define instructions
        List of commands to play game

Define main_game
        Fetch global variable gameOver
        While not gameOver
                Call add_2 function
                Print_board after

Define print_board with parameters board and n
        Fetch global variable score
        For i in range n

For j in range n
Val equals board[i][j]
If val greater than score
Score equals val (updates score)
Print one part of the board

Define create_board with parameter n
Create empty array list board
For i in range n
Append a list of 0's to board list
Return board

Define game_over
Update gameOver variable to true and print score

Define add_2 parameter board and n
Fetch global variable game_over
Inner_count equals 0 (Check if there is any space left to put a new number in)
Create empty lists called zero_x and zero_y
For i and j in range n
If board doesn't equal 0 then increment inner_count
Else append x and y to lists
If inner_count is the whole board
Game_over
If gameOver stop
Variable pos equals random value for 0 spaces
Set board to random zero spot on board to 2

Start game with n = 4

Step 2:

```
import random
score = 0
gameOver = False

def instructions():
   print("The number 2 will appear in random spots and you will need to combine
numbers to save space")
   print("If you run out of room the game ends")
   print("Commands are as follows : ")
   print("Up key or 'w' : Move Up")
   print("Down key or 's' : Move Down")
   print("Left key or 'a' : Move Left")
   print("Right key or 'd' : Move Right")
```

```
    print()

def main_game(n):
    global gameOver
    board = create_board(n)

    while not gameOver:

        add_2(board, n)
        print_board(board,n)
        cmd = input("Enter key: ")
        # move up, down, right, left
        if cmd == 'w': move_up(board,n)
        elif cmd == 's': move_down(board,n)
        elif cmd == 'a': move_left(board,n)
        elif cmd == 'd': move_right(board,n)
        else: print("Incorrect input")

        # print
        print_board(board,n)

def print_board(board,n):
    global score
    for i in range(n):
        for j in range(n):
            val = board[i][j]
            if val > score:
                score = val
            print(f"|{val}|", end = '')
        print()
    print()

def move_up(board,n):
    for i in range(1,n):
        for j in range(n):
            if(board[i-1][j] == 0 and board[i][j] != 0):
                board[i-1][j] = board[i][j]
                board[i][j] = 0
                move_up(board,n)
            elif(board[i-1][j] == board[i][j]):
                board[i-1][j] = board[i][j]*2
                board[i][j] = 0
```

```
def move_down(board,n):
    for i in range(n-1):
        for j in range(n):
            if(board[i+1][j] == 0 and board[i][j] != 0):
                board[i+1][j] = board[i][j]
                board[i][j] = 0
                move_down(board,n)
            elif(board[i+1][j] == board[i][j]):
                board[i+1][j] = board[i][j]*2
                board[i][j] = 0

def move_left(board,n):
    for i in range(n):
        for j in range(1,n):
            if(board[i][j-1] == 0 and board[i][j] != 0):
                board[i][j-1] = board[i][j]
                board[i][j] = 0
                move_left(board,n)
            elif(board[i][j-1] == board[i][j]):
                board[i][j-1] = board[i][j]*2
                board[i][j] = 0

def move_right(board,n):
    for i in range(n):
        for j in range(n-1):
            if(board[i][j+1] == 0 and board[i][j] != 0):
                board[i][j+1] = board[i][j]
                board[i][j] = 0
                move_right(board,n)

            elif(board[i][j+1] == board[i][j]):
                board[i][j+1] = board[i][j]*2
                board[i][j] = 0

def create_board(n):
    board = []
    for i in range(n):
        board.append([0]*n)
    return board

# if all spots on board is taken
```

```
# if all values have no same values above or below
# regression
def game_over():
   global score, gameOver
   gameOver = True
   print(score)

# add a 2 to a random spot with a 0
def add_2(board, n):
   global gameOver
   inner_count = 0

   zero_x =[]
   zero_y =[]

   for i in range(n):
       for j in range(n):
           if(board[i][j] != 0):
               inner_count+=1
           else:
               zero_x.append(i)
               zero_y.append(j)

   if(inner_count == n*n):
       game_over()

   if(gameOver):
       return

   pos = random.randint(0,len(zero_x)-1)
   board[zero_x[pos]][zero_y[pos]] = 2
if __name__ == '__main__':
   instructions()
   main_game(4)
```

Import random

Score is set to 0
gameOver is false

Define instructions
        List of commands to play game

Define main_game
    Fetch global variable gameOver
    Call create_board function save to board variable
    While not gameOver
        Call add_2 function
        Print_board after
        Cmd variable holds user input for next action
        If statements to read which command was entered
            Pass to move_up, down, left or right functions
        Else print incorrect input
        Print board

Define print_board with parameters board and n
    Fetch global variable score
    For i in range n
        For j in range n
            Val equals board[i][j]
            If val greater than score
                Score equals val (updates score)
            Print one part of the board

Define move_up parameters board and n
    For i in range 1 to n (will compare below with above so no need to start from 0)
        For j in range n
            If value above is 0 and value below is not 0 then set bottom value
            And set current value to 0 (swap)
            Call move_up function again to keep moving values upwards (Slide value upwards)
        Else if both bottom and top are equal
            Top will equal double its value and set bottom to 0

Move_down,move_left and move_right are the same as move_up but different direction

Define create_board with parameter n
    Create empty array list board
    For i in range n
        Append a list of 0's to board list
    Return board

Define game_over
    Update gameOver variable to true and print score

Define add_2 parameter board and n

Fetch global variable game_over
Inner_count equals 0 (Check if there is any space left to put a new number in)
Create empty lists called zero_x and zero_y
For i and j in range n
    If board doesn't equal 0 then increment inner_count
    Else append x and y to lists
If inner_count is the whole board
    Game_over
If gameOver stop
Variable pos equals random value for 0 spaces
Set board to random zero spot on board to 2

Start game with n = 4

Extra:
For extra stuff look here:
    https://github.com/DownRamp/Games/blob/master/2048.py
****
THIS IS THE IMPORTANT PART PLEASE DON'T SKIP
****
Next steps:

- Change value to 1 or increase
- Powerups? The clear board once, double current blocks
- Make a more complicated board (https://activityworkshop.net/puzzlesgames/xsudoku/index.html)
-