In [*]:

```
#This module provides access to some variables used or
#maintained by the interpreter and to functions
#that interact strongly with the interpreter.
#It is always available.
import sys
#The OS module in python provides functions for interacting
#with the operating system.
#OS, comes under Python’s standard utility modules.
import os
import random
#The itertools module includes
#a set of functions for working with iterable (sequence-like) data sets.
import itertools

#logic
#1.[2,2]>>>[2,2,0,0]
#2.[2,2]+[0,0,0,0]=[2,2,0,0,0,0]
#3.[2,2,0,0,0,0][:4]=[2,2,0,0]
#slicing
def trim(seqs,direction=0):
    return ([0,0,0,0]+[n for n in seqs if n])[-4:]if direction\
                else ([n for n in seqs if n ]+[0,0,0,0])[:4]

#logic
#same number
#[1,2]
#[0,1][2,3]
def sum_seqs(seqs,direction=0):
    if seqs[1] and seqs[2] and seqs[1]==seqs[2]:
        return trim([seqs[0],seqs[1]*2,0,seqs[3]],\
                    direction=direction)
    if seqs[0] and seqs[1] and seqs[0]==seqs[1]:
        seqs[0],seqs[1]=seqs[0]*2,0
    if seqs[2] and seqs[3] and seqs[2]==seqs[3]:
        seqs[2],seqs[3]=seqs[2]*2,0
    return trim(seqs,direction=direction)
#set up,down,left and right
def up(grid):
    for col in [0,1,2,3]:
        #列举
        #my_list = ['apple', 'banana', 'grapes', 'pear']
        #for c, value in enumerate(my_list, 1):
            #print(c, value)
        # Output:
        # 1 apple
        # 2 banana
        # 3 grapes
        # 4 pear
        #It allows us to loop over something and have an automatic counter.
        for _idx,n in enumerate (sum_seqs(trim([row[col]for row in grid]))):
            grid[_idx][col]=n
    return grid
#add direction=1 at the end for the difference direction of up
def down(grid):
    for col in [0,1,2,3]:
        for _idx,n in enumerate (sum_seqs(trim([row[col]for row in grid],\
                                            direction=1),direction=1)):
                grid[_idx][col]=n
        return grid

def left(grid):
    return [sum_seqs(trim(row)) for row in grid]
#add direction=1, for it different of left
def right (grid):
    return [sum_seqs(trim(row,direction=1),direction=1) for row in grid]

#set  the main game as a class
class Game:
    grid=[]
    controls=['w','a','s','d']
    #the start of the game is have 2 random number from 2 or 4
    #many 2 or 4 is get more chance to have different numbers
    def rnd_field(self):
        number=random.choice([4,2,4,2,4,2,4,2,4,2,4,2])
        x,y=random.choice([(x,y)for x,y in itertools.product\
                                   ([0,1,2,3],[0,1,2,3])if self.grid[x][y]==0])
        self.grid[x][y]=number
    #print screen  use-and|
    #use os.system('cls') one more once it will
    #show the result for the new grid
    #dont need have the one that before the move
    def print_screen(self):
        os.system('cls')
        print('-'*21)
        for row in self.grid:
            print('|',format('|'.join([str(col or'').center(4)\
                                        for col in row])))
            print('_'*21)

    #main logic of the game
    #how to win and how to lost
    def logic(self,control):
        #the way to control the game use wasd
        grid = {'w':up,'a':left,'s':down,'d':right}\
                [control]([[c for c in r] for r in self.grid])
        #the logic why we can win
        if grid!=self.grid:
            del self.grid[:]
            self.grid.extend(grid)
            if [n for n in itertools.chain(*grid) if n>=2048]:
                return 1,'You Win!'
            self.rnd_field()
        else:
            #the logic how can we lost
            if not [1 for g in [f(grid) for f in [up,down,left,right]]\
                      if g == self.grid]:
                return -1,'You Lost'
        return 0,''#1='You Win' 2='You lose'

    def main_loop(self):
        #empty grid
        self.grid=([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])
        self.rnd_field()
        self.rnd_field()
        # now start the game
        while True:
            self.print_screen()
            control=input('Enter w/a/s/d for move the number: ')
            if control in self.controls:
                status,info=self.logic(control)
                if status:
                    print(info)
                    if input('Start another game?[y/n]').lower()=='y':
                        break
                    else:
                        sys.exit(0)
        self.main_loop()
#call the game
if __name__=='__main__':
    Game().main_loop()
```

```
---------------------
|     |    | 2  |
_____________________
|  4  |    |    |
_____________________
|     |    |    |
_____________________
|     |    |    |
_____________________
Enter w/a/s/d for move the number: d
---------------------
|     |    |    | 2
_____________________
|     |    |    | 4
_____________________
|     |    |    |
_____________________
|     |    | 2  |
_____________________
Enter w/a/s/d for move the number: w
---------------------
|     |    | 2  | 2
_____________________
|     |    |    | 4
_____________________
|     |    |    |
_____________________
|     | 2  |    |
_____________________
Enter w/a/s/d for move the number: d
---------------------
|     |    |    | 4
_____________________
|  4  |    |    | 4
_____________________
|     |    |    |
_____________________
|     |    |    | 2
_____________________
```

```
Enter w/a/s/d for move the number:
```

In [ ]: