Basic starting code

```
import random, pygame, sys
from pygame.locals import *

# Define basic variables:
WINDOWWIDTH = 700
WINDOWHEIGHT = 500
FPS = 30
# Plus any others based
# on the game in question:
BOARDWIDTH = 3
BOARDHEIGHT = 3
BOXSIZE = 70
GAPSIZE = 5
REVEALSPEED = 5

# Define margins:
# Margins are made with the following:
# ((full window width/height) - (board width/height *
# (the element width/height + all its padding between blocks))) / 2
XMARGIN = int((WINDOWWIDTH - (BOARDWIDTH * (BOXSIZE + GAPSIZE))) / 2)
YMARGIN = int((WINDOWHEIGHT - (BOARDHEIGHT * (BOXSIZE + GAPSIZE))) / 2)
```

Random use and colors

```
# To choose random from x to y
random.randint( a: 5, WINDOWWIDTH - 6)
# To shuffle array
shuffledList = random.shuffle(list)
# To randomly choose from list:
random.choice(list)
# or
random.choice(range(list)) # for length

# Define colors for use:
#                 R    G    B
BLACK =         (  0,   0,   0)
WHITE =         (255, 255, 255)
RED   =         (255,   0,   0)
GREEN =         (  0, 255,   0)
BLUE  =         (  0,   0, 255)
BROWN =         (150,  75,   0)
BRIGHTBLUE =    (  0,  50, 255)
DARKTURQUOISE = (  3,  54,  73)
```

Sound and picture

```
# Examples
list=['x', 'y']
leftX = 10
topY = 10

# How to define a sound effect
BEEP1 = pygame.mixer.Sound('beep1.ogg')
# To play the sound we just need to:
BEEP1.play()
# Hot to define an image
catImg = pygame.image.load('cat.png')
# To display it we just:
DISPLAYSURF.blit(catImg,  dest: (leftX, topY))
```

Grid dictionaries and lists

```
# How Boards work
# If a board needs to have
# an already predefined layout
MY_GRID = [
    [4, 4, 4],
    [4, 4, 4],
    [4, 4, 4]
]
# To use it in main we can just do this:
# board_copy = [row[:] for row in MY_GRID]

# How Dictionaries work
dictionaryCoords = [{'x': leftX,     'y': topY},
                    {'x': leftX - 1, 'y': topY},
                    {'x': leftX - 2, 'y': topY}]


# list_copy = list[:]
# list_copy = list[from:to]
```

Animations

```
# If we need an animation
def coverBoxesAnimation(board, boxesToCover):
    # Do the "box cover" animation.
    for coverage in range(0, BOXSIZE + REVEALSPEED, REVEALSPEED):
        drawBoxCovers(board, boxesToCover, coverage)
        # In drawBoxCover we just use coverage as the width

def drawBoxCovers(board, boxes, coverage):  1 usage
    # Draws boxes being covered/revealed. "boxes" is a list
    # of two-item lists, which have the x & y spot of the box.
    for box in boxes:
        left, top = getLeftTopOfElement(box[0], box[1])
        pygame.draw.rect(DISPLAYSURF, BGCOLOR, rect:(left, top,
                                                     BOXSIZE, BOXSIZE))
        if coverage > 0: # only draw the cover if there is a coverage
            pygame.draw.rect(DISPLAYSURF, BOXSIZE, rect:(left, top,
                                                         coverage, BOXSIZE))
    pygame.display.update()
    FPSCLOCK.tick(FPS)
```

Make Text Function

```
# Define functino for making text:
def makeText(text, color, bgcolor, left, top):  4 usages
    # create the Surface and Rect objects for some text.
    textSurf = BASICFONT.render(text, antialias True, color, bgcolor)
    textRect = textSurf.get_rect()
    textRect.topleft = (left, top)
    return (textSurf, textRect)
```

```
# To use makeText
# textSurf, textRect = makeText('Message Text',
# TEXTCOLOR, TEXTBGCOLOR(optionally), topPoz, leftPoz)
# DISPLAYSURF.blit(textSurf, textRect)
```

Get **Top Left of box** and get **Box at Pixel**

```
# Define get left top position of a certain element function:
def getLeftTopOfElement(x, y):  4 usages
    left = XMARGIN + (x * (BOXSIZE + GAPSIZE))
    top = YMARGIN + (y * (BOXSIZE + GAPSIZE))
    return left, top

# Define a element at pixel function:
def getBoxAtPixel(mousex, mousey):  1 usage
    for xElement in range(BOARDWIDTH):
        for yElement in range(BOARDHEIGHT):
            left, top = getLeftTopOfElement(xElement, yElement)
            boxRect = pygame.Rect(left, top, BOXSIZE, BOXSIZE)
            if boxRect.collidepoint(mousex, mousey):
                return xElement, yElement
    return None, None
```

Drawing board

```
# Define function to draw a board:
def drawBoard(board):  2 usages
    for xElement in range(BOARDWIDTH):
        for yElement in range(BOARDHEIGHT):
            left, top = getLeftTopOfElement(xElement, yElement)
            if board[xElement][yElement] == BRIGHTBLUE:
                pygame.draw.rect(DISPLAYSURF, BRIGHTBLUE,
                                 rect: (left, top, BOXSIZE, BOXSIZE))
                numberSurf, numberRect = makeText( text: '0', WHITE, BRIGHTBLUE,
                                                  left+int(BOXSIZE/2)-5, top+int(BOXSIZE/2)-7)
                DISPLAYSURF.blit(numberSurf, numberRect)
            else:
                pygame.draw.rect(DISPLAYSURF, GREEN,  rect: (left, top, BOXSIZE, BOXSIZE))
```

Drawing board in cells

```
# In case we need our window to be devided in cells:
def drawGrid():
    for x in range(0, WINDOWWIDTH): # draw vertical lines
        pygame.draw.line(DISPLAYSURF, MESSAGECOLOR,  start_pos: (x, 0),  end_pos: (x, WINDOWHEIGHT))
    for y in range(0, WINDOWHEIGHT): # draw horizontal lines
        pygame.draw.line(DISPLAYSURF, MESSAGECOLOR,  start_pos: (0, y),  end_pos: (WINDOWWIDTH, y))
```

How to **fill empty matrix**

```
def getAllBlueBoard(board):  2 usages
    for x in range(BOARDWIDTH):
        column = []
        for y in range(BOARDHEIGHT):
            column.append(BRIGHTBLUE)
        board.append(column)
```

Terminate

```
# Define function to end the game on exit:
def terminate():  1 usage
    pygame.quit()
    sys.exit()
```

Simple Win Animation

```
# Very basic win animation
def startWinAnimation(board):  1 usage
    color1 = BROWN
    color2 = DARKTURQUOISE
    for i in range(13):
        color1, color2 = color2, color1
        DISPLAYSURF.fill(color1)
        drawBoard(board)
        textSurf, textRect = makeText( text: "You win", WHITE, BLACK,
            int(WINDOWWIDTH / 2) - 40, int(WINDOWHEIGHT / 2) - 40)
        DISPLAYSURF.blit(textSurf, textRect)
        pygame.display.update()
        pygame.time.wait(300)
```

MAIN GAME

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))

    pygame.display.set_caption("My Game")
    BASICFONT = pygame.font.Font( name: None,  size: 30)

    mousex = 0
    mousey = 0

    score = 0

    board = []
    getAllBlueBoard(board)

    while True:
        mouseClicked = False
        drawBoard(board)

        textSurf, textRect = makeText( text: 'DISPLAY MESSAGE', WHITE, BRIGHTBLUE,
                                       int(WINDOWWIDTH / 2) - 100,  top: 20)
        DISPLAYSURF.blit(textSurf, textRect)

        scoreSurf, scoreRect = makeText( text: f'Score: {score}', WHITE, BRIGHTBLUE,
                                         left: 20, WINDOWHEIGHT - 40)
        DISPLAYSURF.blit(scoreSurf, scoreRect)
```

Start main function

```
# Start the main game function
if __name__ == '__main__':
    main()
```

Handling events

```
# Handling events
for event in pygame.event.get():

    # Handling exit game
    if (event.type == pygame.QUIT or
            (event.type == KEYUP and event.key == K_ESCAPE)):
        terminate()

    # Handling mouse events
    elif event.type == MOUSEMOTION:
        mousex, mousey = event.pos
        print(f"mousex = {mousex}, mousey = {mousey}")
    elif event.type == MOUSEBUTTONUP:
        mousex, mousey = event.pos
        mouseClicked = True
        print(f"Mouse button clicked = {mouseClicked}")

    # Handling key button presses
    elif event.type == KEYUP:
        if event.key == K_1:
            # This is how we get the
            # number from the key event
            num = event.key - K_0
```

Common logic and display update

```
xElement, yElement = getBoxAtPixel(mousex, mousey)
# We have to check constantly if we are above a box
if xElement != None and yElement != None:
    highlightBox(xElement, yElement)
    if mouseClicked:
        board[xElement][yElement] = GREEN
        score += 1
        if hasWon(board):
            startWinAnimation(board)
            DISPLAYSURF.fill(BGCOLOR)
            pygame.time.wait(1300)
            board = []
            score = 0
            getAllBlueBoard(board)

pygame.display.update()
FPSCLOCK.tick(FPS)
```