```
import pygame

from pygame.locals import *


pygame.init()


'''

Defining gaming window size and font

'''

Window_width = 500

Window_height = 500


window = pygame.display.set_mode((Window_width, Window_height))

pygame.display.set_caption('Brickstroy')


font = pygame.font.SysFont('Arial', 30)


'''

Defining Bricks colour

'''

O_brick = (255, 100, 10)

w_brick = (255, 255, 255)

g_brick = (0, 255, 0)

black = (0, 0, 0)


game_rows = 6

game_coloumns = 6

clock = pygame.time.Clock()

frame_rate = 60

my_ball = False

game_over = 0

score = 0
```

```
class Ball():
   '''
Creating ball for the game
'''


   def __init__(self, x, y):


      self.radius = 10
      self.x = x - self.radius
      self.y = y - 50
      self.rect = Rect(self.x, self.y, self.radius * 2, self.radius * 2)
      self.x_speed = 4
      self.y_speed = -4
      self.max_speed = 5
      self.game_over = 0


   def motion(self):
      collision_threshold = 5
      block_object = Block.bricks
      brick_destroyed = 1
      count_row = 0
      for row in block_object:
         count_item = 0
         for item in row:
            # check collision with gaming window
            if self.rect.colliderect(item[0]):
               if abs(self.rect.bottom - item[0].top) < collision_threshold and self.y_speed > 0:
                  self.y_speed *= -1


               if abs(self.rect.top - item[0].bottom) < collision_threshold and self.y_speed < 0:
                self.y_speed *= -1
            if abs(self.rect.right - item[0].left) < collision_threshold and self.x_speed > 0:
                self.x_speed *= -1
            if abs(self.rect.left - item[0].right) < collision_threshold and self.x_speed < 0:
                self.x_speed *= -1

            if block_object[count_row][count_item][1] > 1:
                block_object[count_row][count_item][1] -= 1
            else:
                block_object[count_row][count_item][0] = (0, 0, 0, 0)

        if block_object[count_row][count_item][0] != (0, 0, 0, 0):
            brick_destroyed = 0
        count_item += 1
    count_row += 1

if brick_destroyed == 1:
    self.game_over = 1

# check for collision with bricks
if self.rect.left < 0 or self.rect.right > Window_width:
    self.x_speed *= -1

if self.rect.top < 0:
    self.y_speed *= -1
if self.rect.bottom > Window_height:
    self.game_over = -1

# check for collission with base
if self.rect.colliderect(user_basepad):
    if abs(self.rect.bottom - user_basepad.rect.top) < collision_threshold and self.y_speed > 0:
        self.y_speed *= -1
        self.x_speed += user_basepad.direction
        if self.x_speed > self.max_speed:
            self.x_speed = self.max_speed
        elif self.x_speed < 0 and self.x_speed < -self.max_speed:
            self.x_speed = -self.max_speed
        else:
            self.x_speed *= -1

self.rect.x += self.x_speed
self.rect.y += self.y_speed

return self.game_over

def draw(self):
pygame.draw.circle(window, (0, 0, 255), (self.rect.x +
                self.radius, self.rect.y + self.radius), self.radius)
pygame.draw.circle(window, (255, 255, 255), (self.rect.x +
                self.radius, self.rect.y + self.radius), self.radius, 1)

def reset(self, x, y):

self.radius = 10
self.x = x - self.radius
self.y = y - 50
self.rect = Rect(self.x, self.y, self.radius * 2, self.radius * 2)
self.x_speed = 4
self.y_speed = -4
self.max_speed = 5
self.game_over = 0
```

```
class Block():
    '''
This class will help me create Blocks/bricks of the game
'''


    def __init__(self):
        self.width = Window_width // game_coloumns
        self.height = 40


    def make_brick(self):
        self.bricks = []
        single_brick = []
        for row in range(game_rows):

            brick_row = []

            for coloumn in range(game_coloumns):

                x_brick = coloumn * self.width
                y_brick = row * self.height
                rect = pygame.Rect(x_brick, y_brick, self.width, self.height)
                # assign power to the bricks based on row
                if row < 2:
                    power = 3
                elif row < 4:
                    power = 2
                elif row < 6:
                    power = 1

                single_brick = [rect, power]
            brick_row.append(single_brick)

          self.bricks.append(brick_row)

    def draw_brick(self):
        for row in self.bricks:
            for brick in row:

                if brick[1] == 3:
                    brick_colour = O_brick
                elif brick[1] == 2:
                    brick_colour = w_brick
                elif brick[1] == 1:
                    brick_colour = g_brick
                pygame.draw.rect(window, brick_colour, brick[0])
                pygame.draw.rect(window, black, (brick[0]), 1)

class base():
    '''
This class is to create the base pad of the game
'''

    def __init__(self):

        self.height = 20
        self.width = int(Window_width / game_coloumns)
        self.x = int((Window_width / 2) - (self.width / 2))
        self.y = Window_height - (self.height * 2)
        self.speed = 8
        self.rect = Rect(self.x, self.y, self.width, self.height)
        self.direction = 0
    def slide(self):


        self.direction = 0
        key = pygame.key.get_pressed()
        if key[pygame.K_LEFT] and self.rect.left > 0:
            self.rect.x -= self.speed
            self.direction = -1
        if key[pygame.K_RIGHT] and self.rect.right < Window_width:
            self.rect.x += self.speed
            self.direction = 1

    def draw(self):
        pygame.draw.rect(window, (0, 0, 255), self.rect)
        pygame.draw.rect(window, (255, 255, 255), self.rect, 1)

    def reset(self):


        self.height = 20
        self.width = int(Window_width / game_coloumns)
        self.x = int((Window_width / 2) - (self.width / 2))
        self.y = Window_height - (self.height * 2)
        self.speed = 8
        self.rect = Rect(self.x, self.y, self.width, self.height)
        self.direction = 0

def draw_text(text, font, w_brick, x, y):
    '''
    Funtion for showing text in gaming window
    '''
    image = font.render(text, True, w_brick)
```

```
   window.blit(image, (x, y))


Block = Block()
# Creating Brick
Block.make_brick()
# Defining base pad
user_basepad = base()
ball = Ball(user_basepad.x + (user_basepad.width // 2),
          user_basepad.y - user_basepad.height)   # Defining ball


game = True
while game:


   clock.tick(frame_rate)
   window.fill(black)                      # Gaming window Background
   Block.draw_brick()                      # Drawing bricks
   user_basepad.draw()                     # Drawing user basepad
   ball.draw()                             # Drawing gaming ball


   if my_ball:
      user_basepad.slide()
      game_over = ball.motion()
      if game_over != 0:
         my_ball = False


   # Game Info on the gaming window
   if not my_ball:
      if game_over == 0:
         draw_text('CLICK ANYWHERE TO START', font,
                  w_brick, 90, Window_height // 2 + 100)
      elif game_over == 1:
            draw_text('YOU WON!', font, w_brick, 180, Window_height // 2 + 50)
            draw_text('CLICK ANYWHERE TO RESTART', font,
                      w_brick, 90, Window_height // 2 + 100)
        elif game_over == -1:
            draw_text('GAME OVER!', font, w_brick,
                      180, Window_height // 2 + 50)
            draw_text('CLICK ANYWHERE TO RESTART', font,
                      w_brick, 90, Window_height // 2 + 100)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game = False
        if event.type == pygame.MOUSEBUTTONDOWN and my_ball == False:
            my_ball = True
            ball.reset(user_basepad.x + (user_basepad.width // 2),
                       user_basepad.y - user_basepad.height)
            user_basepad.reset()
            Block.make_brick()

    pygame.display.update()

pygame.quit()
```