```
# To run this game install pygame with `pip install pygame`

import pygame
import random
from time import sleep

class Ball():
    """Class to define a ball"""

    def __init__(self, screen_width, screen_height):
        self.size = 10
        self.x = random.randint(0, screen_width * 0.8)
        self.y = random.randint(0, screen_height * 0.8)
        self.x_vel = 1
        self.y_vel = 1
        self.screen_width = screen_width
        self.screen_height = screen_height
        return


    def detect_collision(self, paddle_y, paddle_height, paddle_width):
        """"Method to detect screen edge, detetct losing,
        and detect hitting the paddle """

        lose = 0
        hit = 0

        # Check if ball has hit the left hand side of the screen
        if self.x <= 0:
            # Mark as lose (will be over-ridden if ball hits paddle)
            lose = 1

        # Check if ball has hit the right hand side of the screen
        if self.x + self.size >= self.screen_width:
            # Reverse x velcoity
            self.x_vel = -self.x_vel

        # Check if ball has hit top or bottom of screen
        if self.y <=0 or self.y + self.size >= self.screen_height:
            # Reverse y velocity
            self.y_vel = -self.y_vel

        # Paddle collision detection

        ball_centre_y = self.y + (self.size/2)

        if (self.x <= paddle_width and ball_centre_y > paddle_y and
                ball_centre_y < paddle_y + paddle_height):

            # Ball hits paddle. Record and reverse x velocity
            lose = 0
            hit = 1
            self.x_vel *= -1

            # identify region of paddle hit (and adjust y velocity)
            region_fraction = (ball_centre_y - paddle_y) / paddle_height
            if region_fraction <= 0.25:
                self.y_vel = -2
            elif region_fraction <= 0.5:
                self.y_vel = -1
            elif region_fraction <= 0.75:
                self.y_vel = 1
            else:
                self.y_vel = 2
```

```
        # Return if game lost or if paddle has hit ball
        return lose, hit


    def move_ball(self, paddle_y, paddle_height, paddle_width):
        """Method to move ball"""

        # Move ball
        self.x += self.x_vel
        self.y += self.y_vel

        # Check for collision with screen edges or paddle
        lose = self.detect_collision(paddle_y, paddle_height, paddle_width)

        return lose

class Game():
    """
    Class to initiaite and run game
    """

    def __init__(self):
        """
        Constructor method for game
        """

        # Initialise pygame
        pygame.init()

        # Set initial lives
        self.lives = 3

        # Hit counter (number of times the paddle hits the ball)
        self.hits = 0

        # Set delay between game loops (will reduce during the game)
        self.delay = int(5)

        # Set text font
        # Use pygame.font.get_fonts() after pygame.init() to see avilable fonts
        self.font = pygame.font.SysFont('dejavusansmono', 18)

        # Set window width and height
        self.screen_width = 800
        self.screen_height = 600

        # Initiate pygame window
        self.win = (pygame.display.set_mode(
            (self.screen_width, self.screen_height)))
        pygame.display.set_caption('Moving blocks')

        # Repeat game while lives remain
        while self.lives > 0:

            # Initialise ball
            self.ball = Ball(self.screen_width, self.screen_height)

            # Initialise paddle
            self.paddle = Paddle(self.screen_width, self.screen_height)

            # Initiate game loop (ends with life lost)
            self.continue_loop = True
            self.loop_game()
```

```
            # Once game loop is finished subtract a life
            self.lives -= 1

            # Call display to show when miss
            self.display_on_miss()

        # Quit game
        pygame.quit()

        return


    def check_events(self):
        """
        Check for close game window
        """
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()

        return


    def display_on_miss(self):
        # Clear screen
        self.win.fill((0, 0, 0))

        # Display message if lives are left
        if self.lives > 0:
            text_string = 'You missed! Remiaining lives: ' + str (self.lives)
            self.win.blit(self.font.render(
                text_string, True, (255,0,0)), (250, 250))

        # Display message if no lives left
        else:
            text_string = 'Game over. Number of hits: ' + str(self.hits)
            self.win.blit(self.font.render(
                text_string, True, (255,0,0)), (250, 250))

        # Render new screen
        pygame.display.update()

        # Display for two seconds before continuing
        sleep (2)

        # Display message for all
        text_string = 'Press any key to continue.'
        self.win.blit(self.font.render(
            text_string, True, (255,0,0)), (250, 280))

        # Render new screen
        pygame.display.update()

        # Wait for key press
        pygame.event.clear()
        pygame.event.wait()

        return


    def loop_game(self):
        """
        Main game loop
```

```
        """

        # Reset game loop delay (controls speed of movement)
        self.delay = int(5)

        # Record nmber of hits in this game (used to increase game speed)
        self.hits_this_ball = 0

        while self.continue_loop:

            # Loop delay (longer delay leads to slower movement)
            pygame.time.delay(self.delay)

            # Check events (for game close)
            self.check_events()

            # Clear screen to all black
            self.win.fill((0, 0, 0))

            # Move ball (and check for miss otr hit with paddle)
            miss, hit = self.ball.move_ball(
                self.paddle.y, self.paddle.height, self.paddle.width)

            # Increment hits and reduce loop pause every 5 hits
            self.hits += hit
            self.hits_this_ball += hit
            if hit ==1 and self.hits_this_ball % 5 == 0 and self.delay > 0:
                self.delay -= 1

            # Set loop to stop if ball has gone out of play
            if miss == 1:
                self.continue_loop = False

            # Move paddle
            self.paddle.move()

            # Redraw ball
            pygame.draw.rect(self.win,(255, 255, 255),
                (self.ball.x, self.ball.y, self.ball.size, self.ball.size))

            # Redraw paddle
            pygame.draw.rect(self.win, (255, 255, 255),
                (self.paddle.x, self.paddle.y,
                self.paddle.width, self.paddle.height))

            # Display lives left and hits (top right of screen)

            text_string = 'Lives left: ' + str (self.lives)
            self.win.blit(
                self.font.render(
                    text_string, True, (255,0,0)), (650, 15))

            text_string = 'Hits: ' + str (self.hits)
            self.win.blit(
                self.font.render(
                    text_string, True, (255,0,0)), (650, 35))

            # Render new screen
            pygame.display.update()

        return

class Paddle():
    """Paddle class"""
```

```
    def __init__(self, screen_width, screen_height):
        self.width = 5
        self.height = 100
        self.screen_width = screen_width
        self.screen_height = screen_height
        self.x = 0
        self.y = int(screen_height/2)
        self.velocity = 3

        return

    def move(self):
        """Move paddle"""
        # Move paddle if key is held down
        keys = pygame.key.get_pressed()

        if keys[pygame.K_UP] and self.y > 0:
            self.y -= self.velocity

        if keys[pygame.K_DOWN] and self.y + self.height < self.screen_height:
            self.y += self.velocity

        return

game = Game()
```