```
Import pygame
Import random
Import time

Pygame.init()

Clock = pygame.time.Clock()
Fps = 60

# Game window
Tile_size = 50
Cols = 20
Margin = 100
Screen_width = tile_size * cols
Screen_height = (tile_size * cols) + margin

Screen = pygame.display.set_mode((screen_width, screen_height))
Pygame.display.set_caption('Rail Rush Clone')

# Function to load images with error handling
Def load_image(path, size=None):
  Try:
    Image = pygame.image.load(path)
    If size:
      Image = pygame.transform.scale(image, size)
    Return image
```

```
    Except pygame.error as e:
        Print(f”Unable to load image at {path}. Error: {e}”)
        Return None

# Load images
Bg_img = load_image(r’C:\Users\sheeb\OneDrive\Desktop\Documents\bg.jpg’, (screen_width, screen_height – margin))
Wood_img = load_image(r’C:\Users\sheeb\OneDrive\Desktop\Documents\obstacle.jpg’, (tile_size, tile_size))
Banana_img = load_image(r’C:\Users\sheeb\OneDrive\Desktop\Documents\banana.jpg’, (tile_size, tile_size))
Monkey_img = load_image(r’C:\Users\sheeb\OneDrive\Desktop\Documents\monkey.jpg’, (tile_size * 2, int(tile_size * 3)))

# Define game variables
Score = 0
Gravity = 1
Game_over = False

# Define colors
White = (255, 255, 255)
Green = (144, 201, 120)
Black = (0, 0, 0)

Font = pygame.font.SysFont(‘Futura’, 24)

# Player class
```

```
Class Player():
    Def __init__(self, x, y):
        Self.rect = pygame.Rect(x, y, tile_size * 2, int(tile_size * 3))
        Self.vel_y = 0
        Self.jump_power = 15
        Self.jump_count = 0
        Self.max_jumps = 3  # Max number of times the up arrow can be pressed to increase the jump height
        Self.in_air = True

    Def move(self):
        Dx = 0
        Dy = 0

        # Get key presses
        Key = pygame.key.get_pressed()
        If key[pygame.K_UP] and self.jump_count < self.max_jumps:
            If not self.in_air:
                Self.vel_y = -self.jump_power
                Self.in_air = True
            Else:
                Self.vel_y = -self.jump_power * (self.max_jumps – self.jump_count)  # Increase jump height
            Self.jump_count += 1
        If key[pygame.K_LEFT]:
            Dx = -5
        If key[pygame.K_RIGHT]:
```

```
        Dx = 5

    # Gravity
    Self.vel_y += gravity
    If self.vel_y > 10:
        Self.vel_y = 10
    Dy += self.vel_y

    # Update player position
    Self.rect.x += dx
    Self.rect.y += dy

    # Collision with ground
    If self.rect.bottom > screen_height – margin:
        Self.rect.bottom = screen_height – margin
        Dy = 0
        Self.in_air = False
        Self.jump_count = 0  # Reset jump count when on the ground

    # Draw the player
    Screen.blit(monkey_img, self.rect.topleft)

    Return dx, dy

# Obstacle class
Class Obstacle(pygame.sprite.Sprite):
```

```
    Def __init__(self, x, y):
        Pygame.sprite.Sprite.__init__(self)
        Self.image = wood_img
        Self.rect = self.image.get_rect()
        Self.rect.x = x
        Self.rect.y = y

    Def update(self):
        Self.rect.x -= 5
        If self.rect.right < 0:
            Self.kill()

# Banana class
Class Banana(pygame.sprite.Sprite):
    Def __init__(self, x, y):
        Pygame.sprite.Sprite.__init__(self)
        Self.image = banana_img
        Self.rect = self.image.get_rect()
        Self.rect.x = x
        Self.rect.y = y

    Def update(self):
        Self.rect.y += 5  # Falling speed
        If self.rect.top > screen_height:
            Self.kill()
```

```
# Create sprite groups
Obstacle_group = pygame.sprite.Group()
Banana_group = pygame.sprite.Group()
Player = Player(screen_width // 2 – tile_size, screen_height – margin – tile_size * 1.5)

# Function to draw text
Def draw_text(text, font, text_col, x, y):
   Img = font.render(text, True, text_col)
   Screen.blit(img, (x, y))

# Timer variables
Last_banana_time = time.time()
Obstacle_spawn_interval = 15  # 15 bananas

# Main game loop
Run = True
Banana_counter = 0

While run:
   Clock.tick(fps)

   # Draw background
   Screen.fill(green)
   If bg_img:
      Screen.blit(bg_img, (0, 0))
```

```
# Draw black road in the center
Road_width = screen_width // 4
Road_height = tile_size
Road_x = (screen_width – road_width) // 2
Road_y = screen_height – margin – tile_size
Pygame.draw.rect(screen, black, (road_x, road_y, road_width, road_height))

# Draw player
Dx, dy = player.move()

# Update and draw groups
Obstacle_group.update()
Obstacle_group.draw(screen)
Banana_group.update()
Banana_group.draw(screen)

# Check for collision with obstacles
If pygame.sprite.spritecollide(player, obstacle_group, False):
   Game_over = True

# Check for collision with bananas
If pygame.sprite.spritecollide(player, banana_group, True):
   Score += 1
   Banana_counter += 1
   Last_banana_time = time.time()  # Update timer when banana is caught
```

```
# Draw the score
Draw_text(f'Score: {score}', font, white, tile_size, screen_height – 60)

# Generate new bananas
If random.randint(1, 50) == 1:
   Banana_group.add(Banana(random.randint(0, screen_width – tile_size), -tile_size))

# Check if it's time to spawn a new obstacle
If banana_counter >= obstacle_spawn_interval:
   Obstacle_group.add(Obstacle(screen_width, screen_height – margin – tile_size * 2))
   Banana_counter = 0  # Reset the banana counter after spawning an obstacle

# Event handler
For event in pygame.event.get():
   # Quit game
   If event.type == pygame.QUIT:
      Run = False

If game_over:
   Draw_text('Game Over', font, white, screen_width // 2 – 100, screen_height // 2)
   Pygame.display.update()
   Pygame.time.wait(2000)
   Run = False

# Update game display window
Pygame.display.update()
```

Pygame.quit()