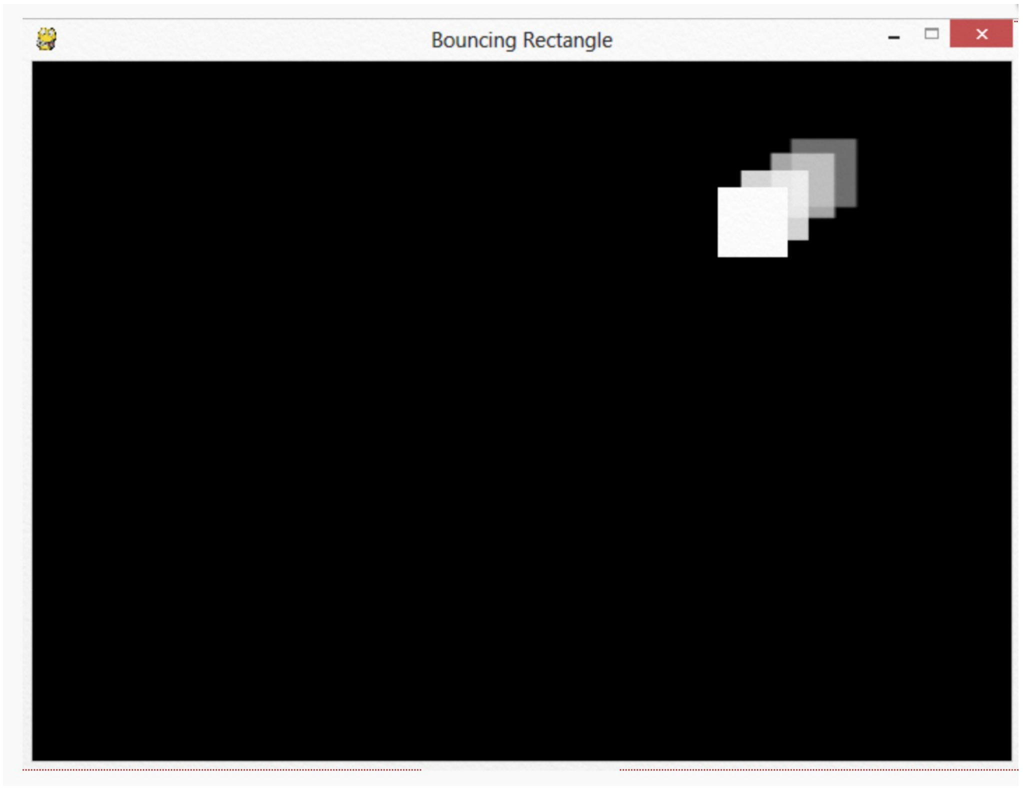
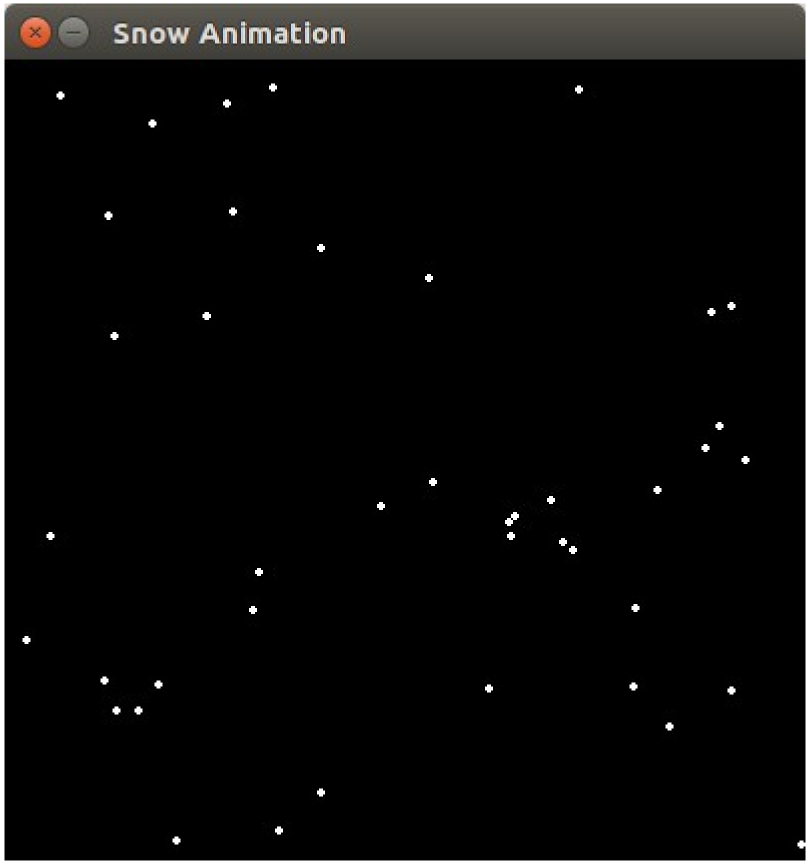
Animation – Bouncing Square & Snow Flakes

- Square that bounces around inside window
- Falling snow





Who am I?

- Tim Hood
- Software Engineer, Computer Programmer, Coder, Developer
- BAE Systems in Yeovil, ~300 people
- Coding aircraft, ship, communications, website and mobile phone software for 30 years

Copy my code and change it

- Copy **bounce.py** and **snow.py** from :-

  “R:\Curriculum\Computing\Yr_8\Y8U4_Programming with Python”

- And save the files here :-

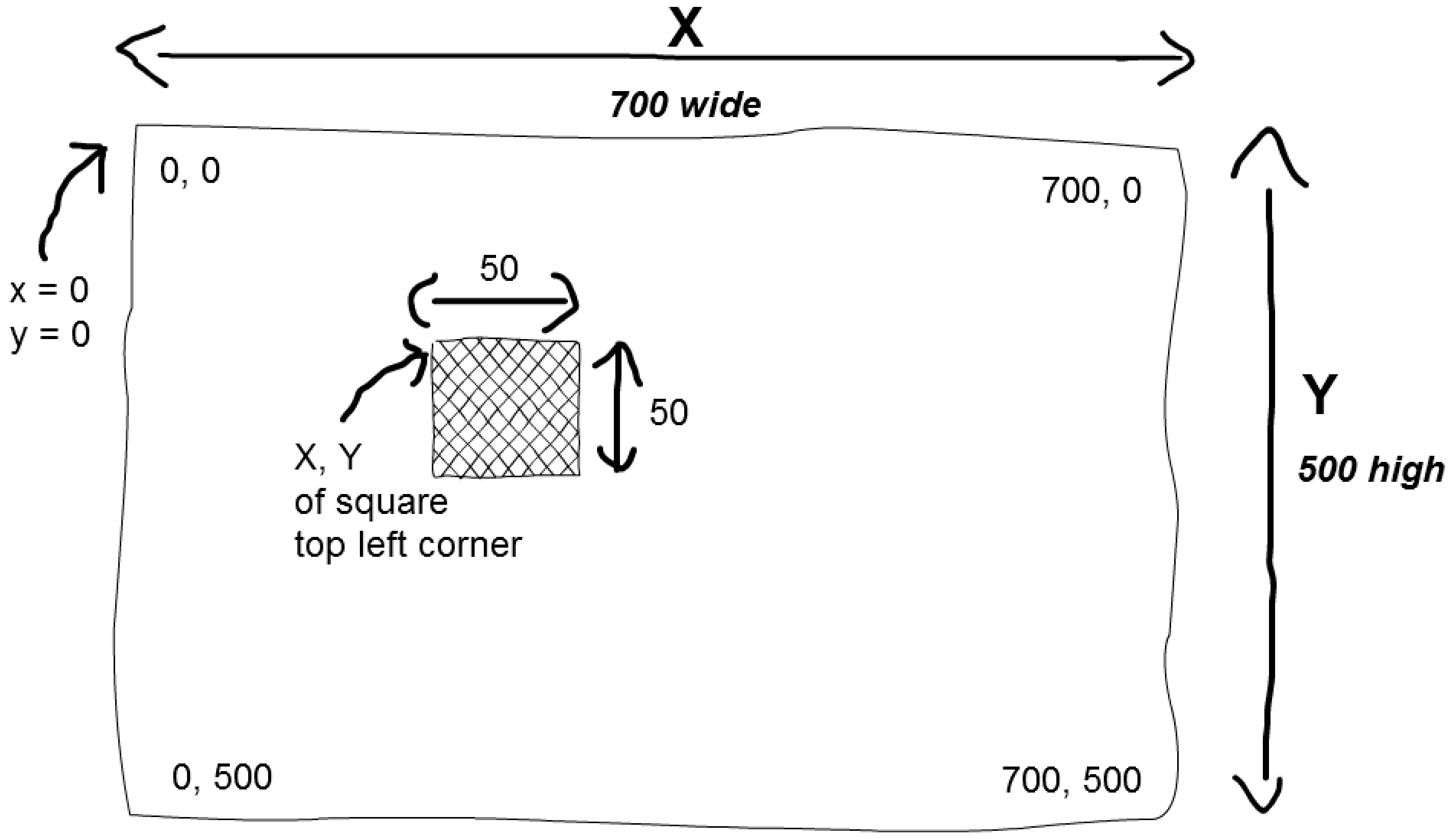
  “C:\temp”

- Open **bounce.py** with WingIDE

Bouncing Square

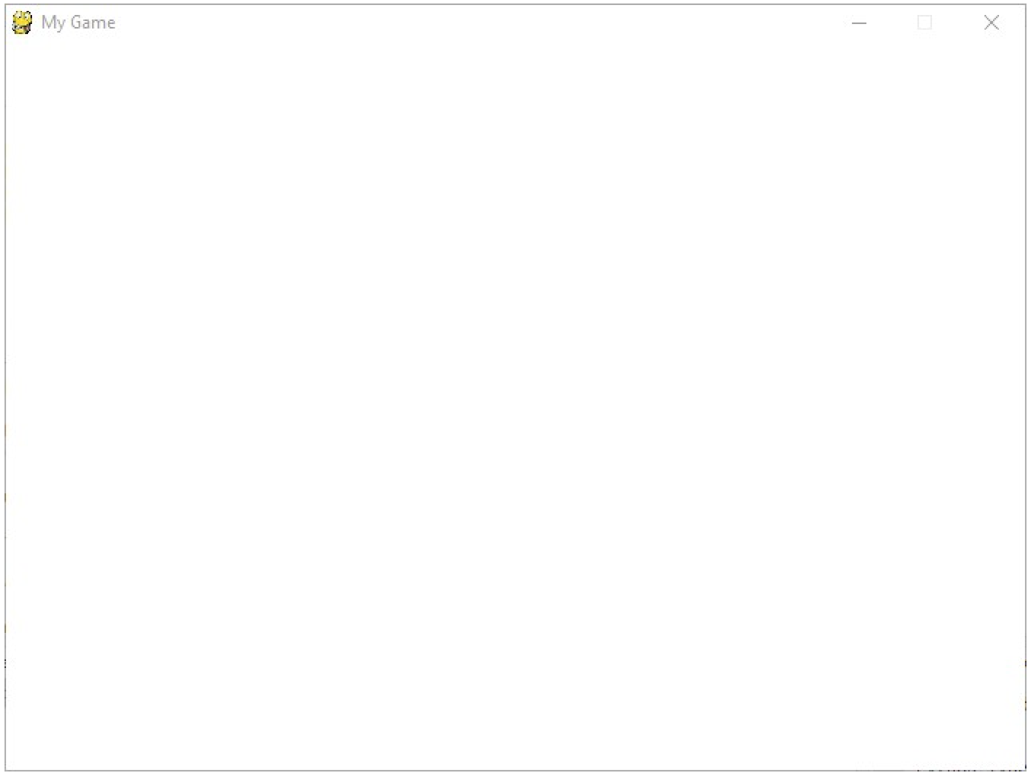
- We will change your copy of **bounce.py**
- You need to find the code and change it
- 6 stages …
  - Display basic window, 700 x 500
  - Black window with white square, 50 x 50
  - Move square to the right
  - Move down and right
  - Bounce when hit edges of window
  - Add inner red square

X and Y coordinates of screen



Run your copy of **bounce.py**

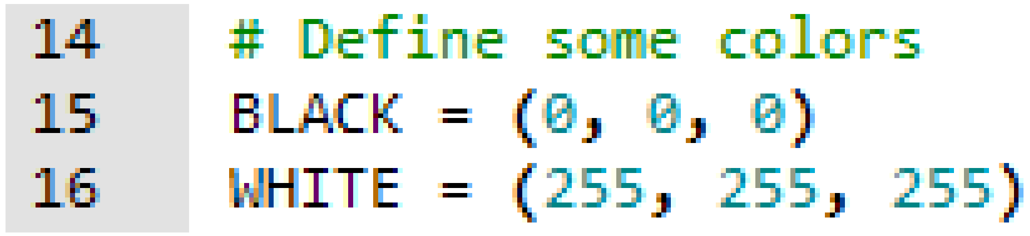
- Open file in WingIDE and run it get a blank white window



- Close window click [X] in corner

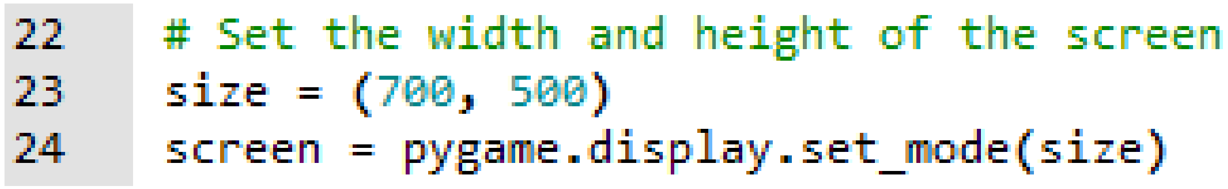
What does the code do?

- Lines 15 - 16
  define some
  colours



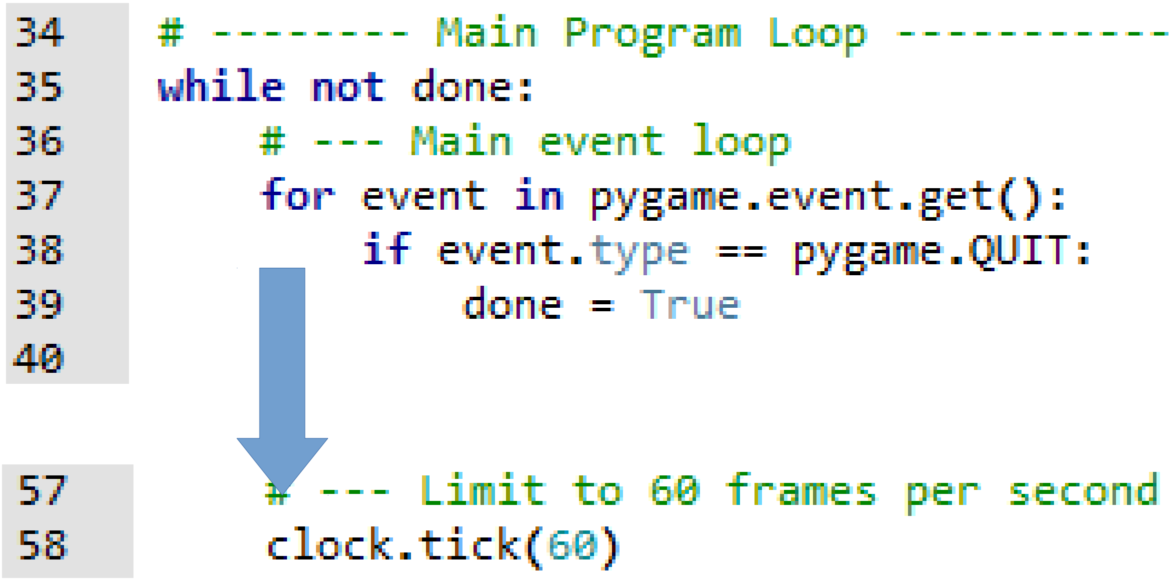
```
14    # Define some colors
15    BLACK = (0, 0, 0)
16    WHITE = (255, 255, 255)
```

- Line 23
  sets size of the
  window
  width, height



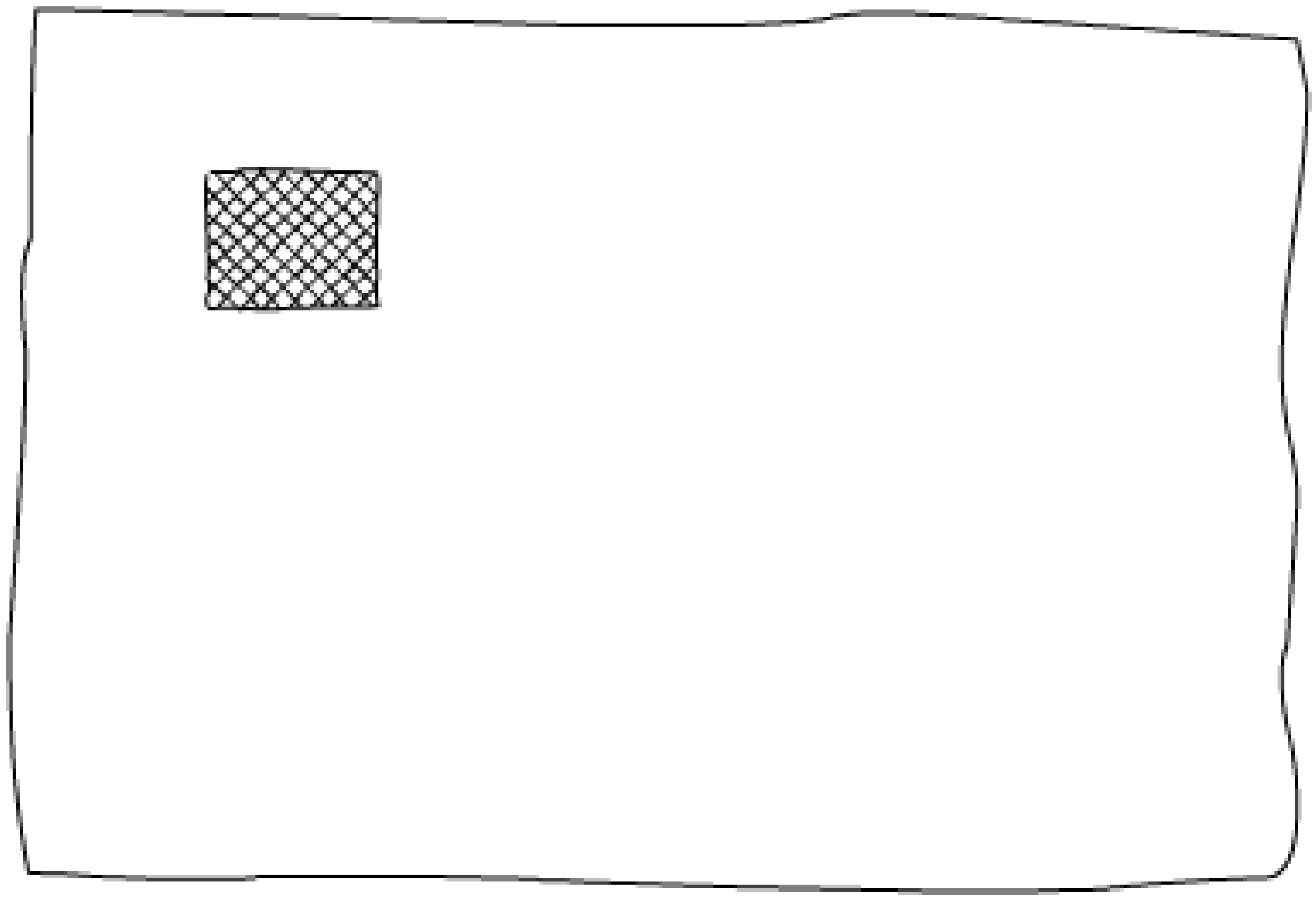
```
22    # Set the width and height of the screen
23    size = (700, 500)
24    screen = pygame.display.set_mode(size)
```

- Lines 35 – 58
  loops until you
  click the [X]



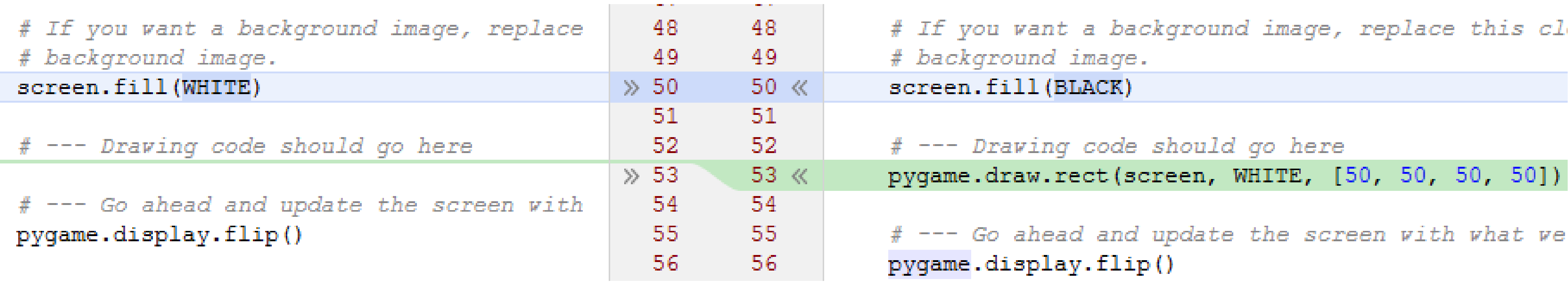
```
34    # -------- Main Program Loop -----------
35    while not done:
36        # --- Main event loop
37        for event in pygame.event.get():
38            if event.type == pygame.QUIT:
39                done = True
40

57        # --- Limit to 60 frames per second
58        clock.tick(60)
```

Draw the white square



Draw the white square

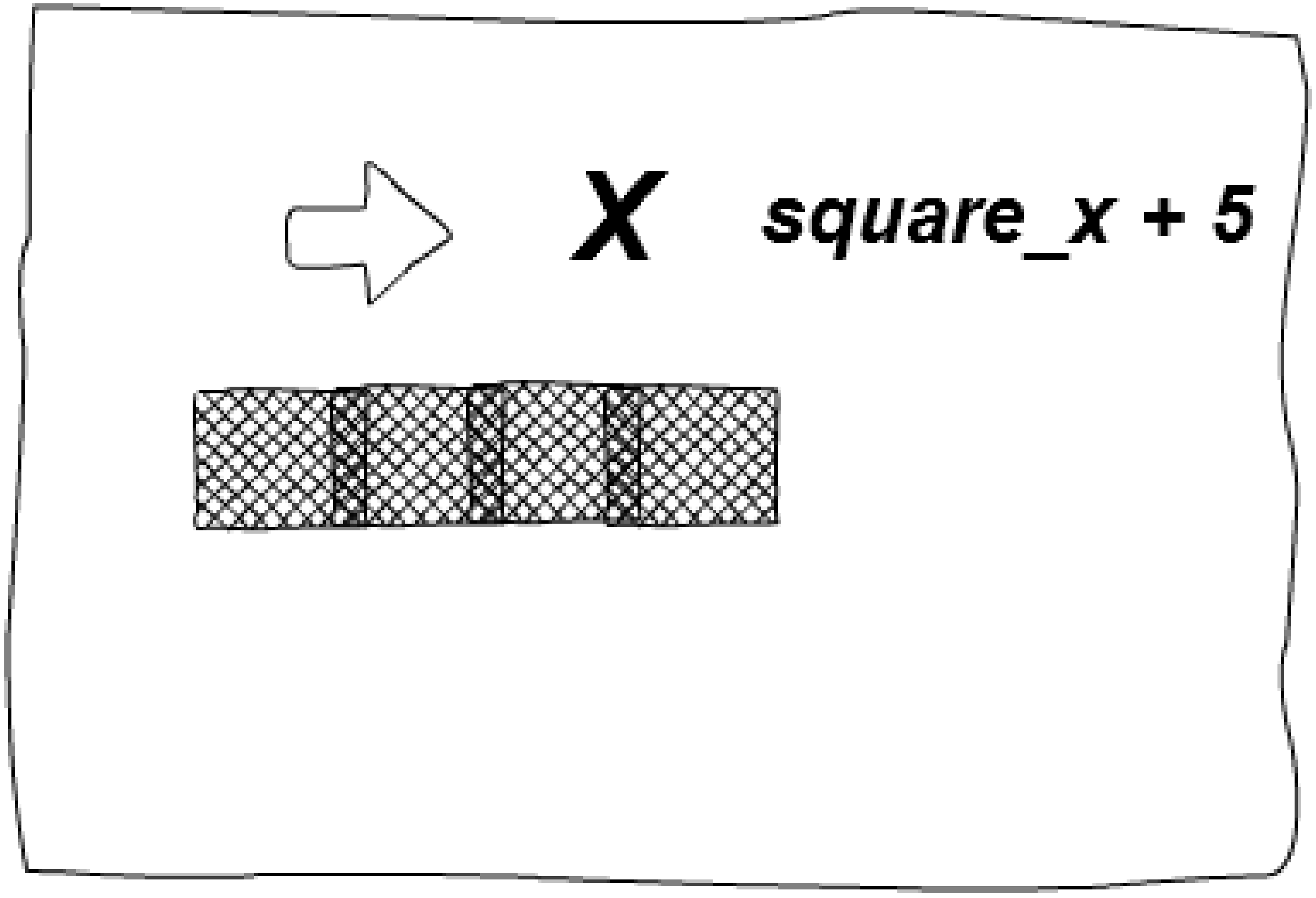
- Change the code highlighted below
  - Background changes from WHITE to BLACK
  - Insert new line to draw 'square'
    - Square is actually a WHITE rectangle
      - [50, 50, 50, 50] is [x, y, width, height]



```
# If you want a background image, replace
# background image.
screen.fill(WHITE)

# --- Drawing code should go here

# --- Go ahead and update the screen with
pygame.display.flip()
```

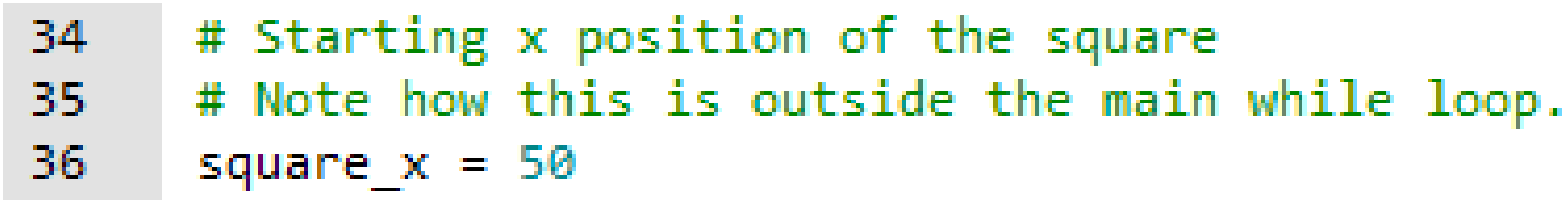
```
# If you want a background image, replace this cl
# background image.
screen.fill(BLACK)

# --- Drawing code should go here
pygame.draw.rect(screen, WHITE, [50, 50, 50, 50])

# --- Go ahead and update the screen with what we
pygame.display.flip()
```

Move the square to the right



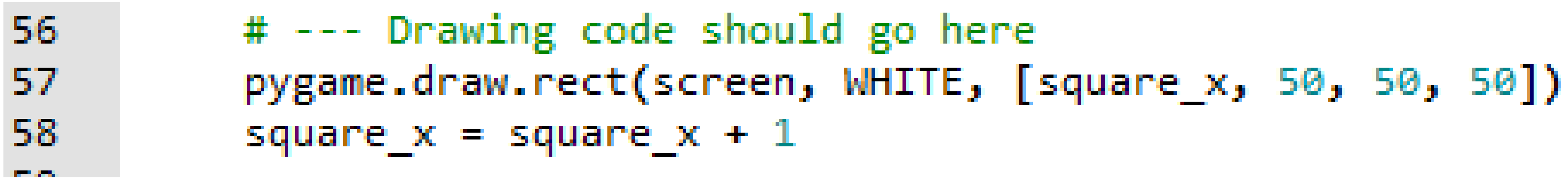
Move the square to the right

- Variable 'square_x' above while loop is 'x' coordinate of square



```
# Starting x position of the square
# Note how this is outside the main while loop.
square_x = 50
```

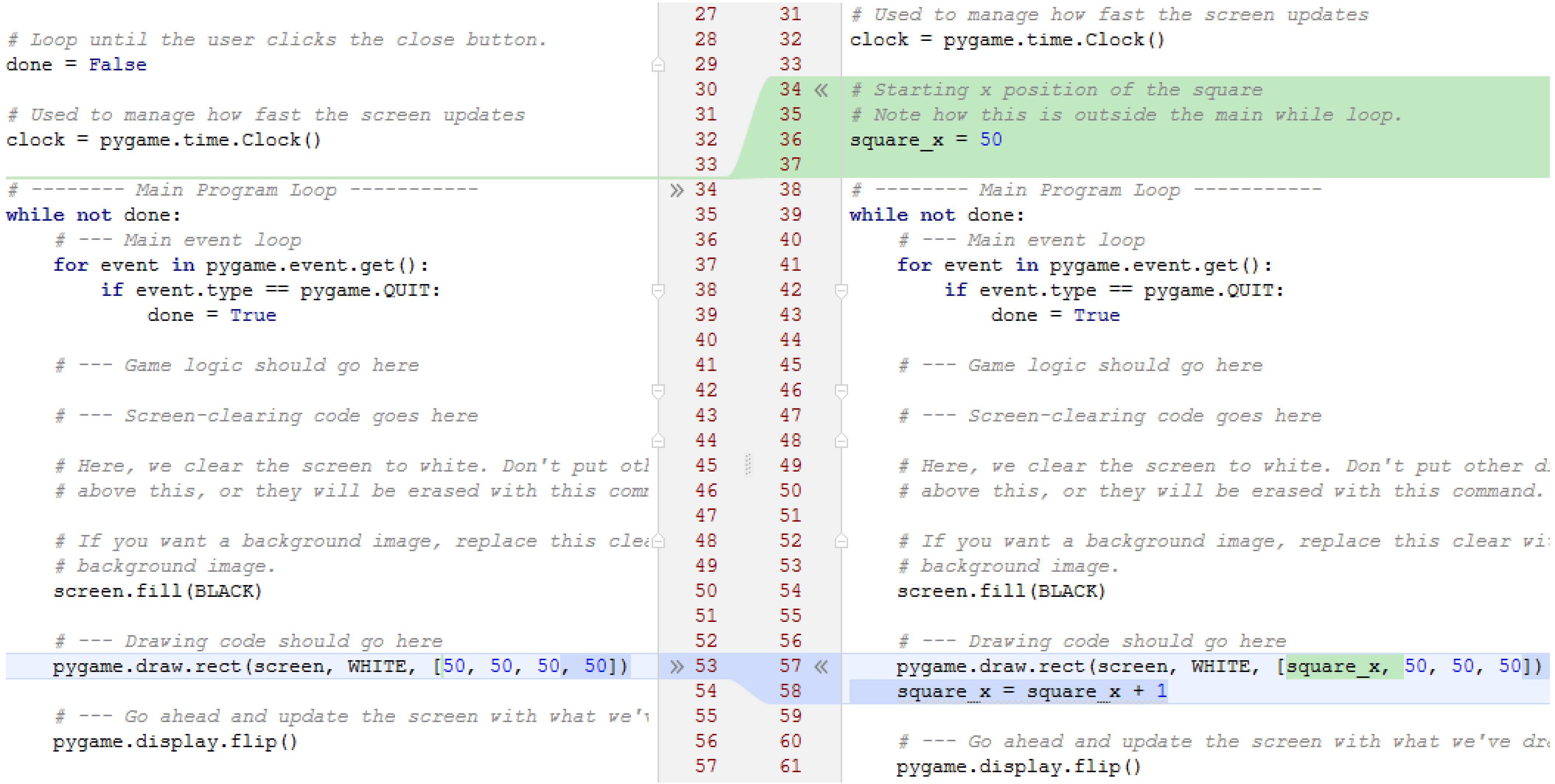
- Use 'square_x' when draw the 'square'



```
    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [square_x, 50, 50, 50])
    square_x = square_x + 1
```

- Increase 'square_x' inside the loop by 1 slow, if increase by 5 goes faster

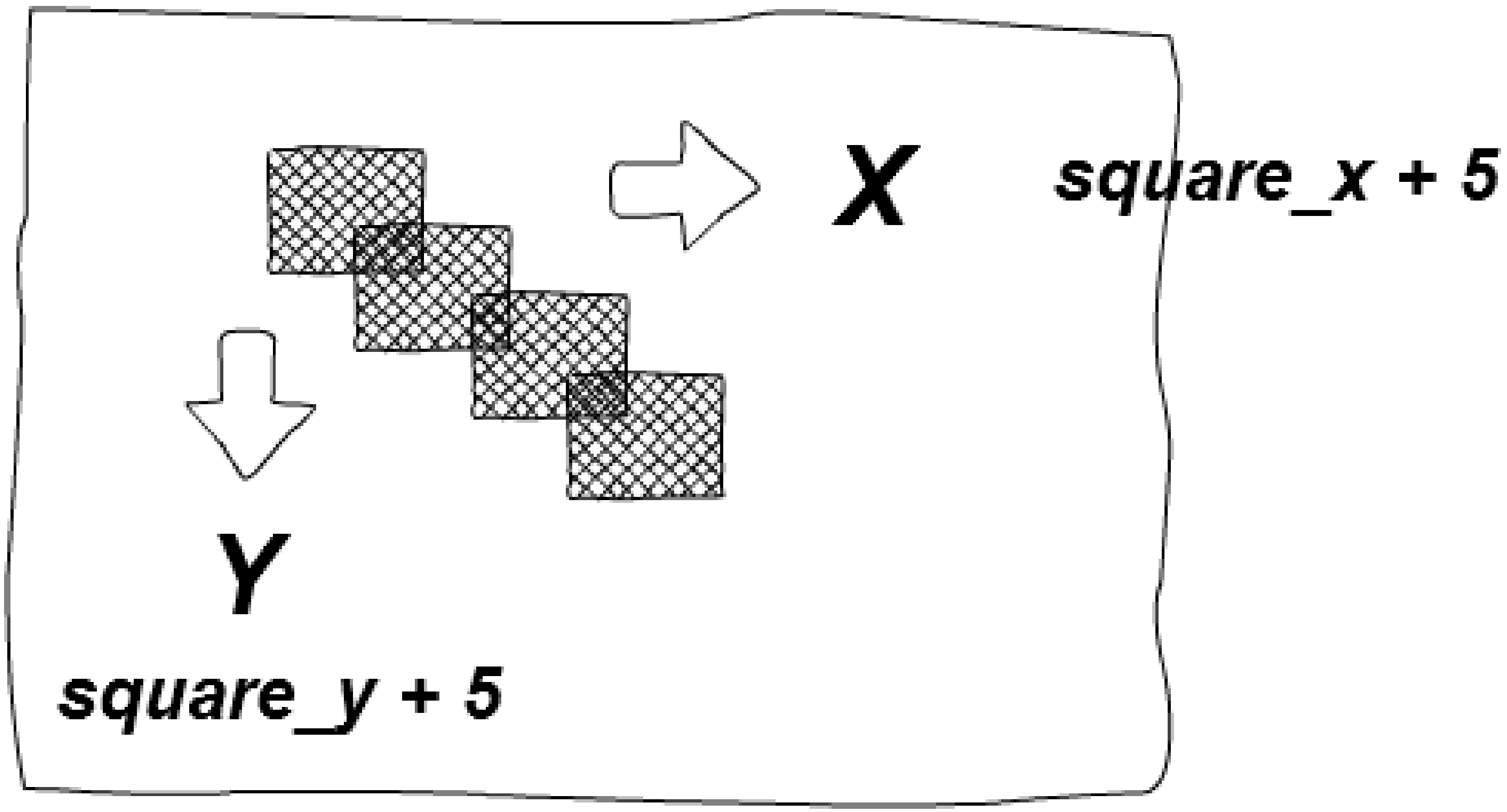
Move the square to the right



```python
# Loop until the user clicks the close button.
done = False

# Used to manage how fast the screen updates
clock = pygame.time.Clock()

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # --- Game logic should go here

    # --- Screen-clearing code goes here

    # Here, we clear the screen to white. Don't put otl
    # above this, or they will be erased with this comr

    # If you want a background image, replace this clea
    # background image.
    screen.fill(BLACK)

    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [50, 50, 50, 50])

    # --- Go ahead and update the screen with what we'
    pygame.display.flip()
```

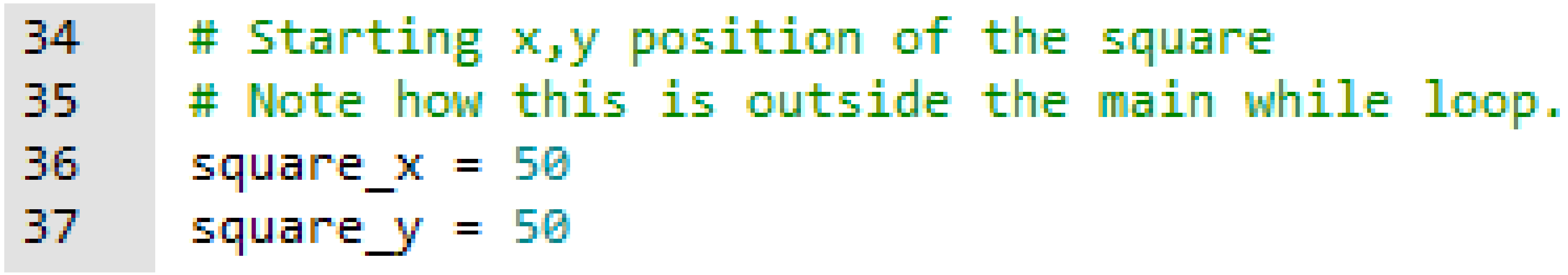
```python
# Used to manage how fast the screen updates
clock = pygame.time.Clock()

# Starting x position of the square
# Note how this is outside the main while loop.
square_x = 50

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # --- Game logic should go here

    # --- Screen-clearing code goes here

    # Here, we clear the screen to white. Don't put other d
    # above this, or they will be erased with this command.

    # If you want a background image, replace this clear wi
    # background image.
    screen.fill(BLACK)

    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [square_x, 50, 50, 50])
    square_x = square_x + 1

    # --- Go ahead and update the screen with what we've dr
    pygame.display.flip()
```

Move the square down & to the right



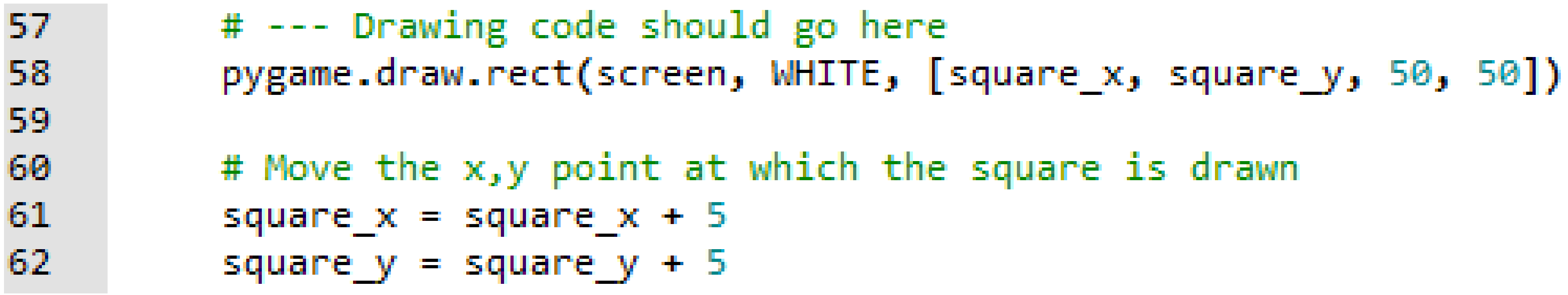
Move the square down & to the right

- Variable 'square_y' above while loop is 'y' coordinate of square



```
# Starting x,y position of the square
# Note how this is outside the main while loop.
square_x = 50
square_y = 50
```

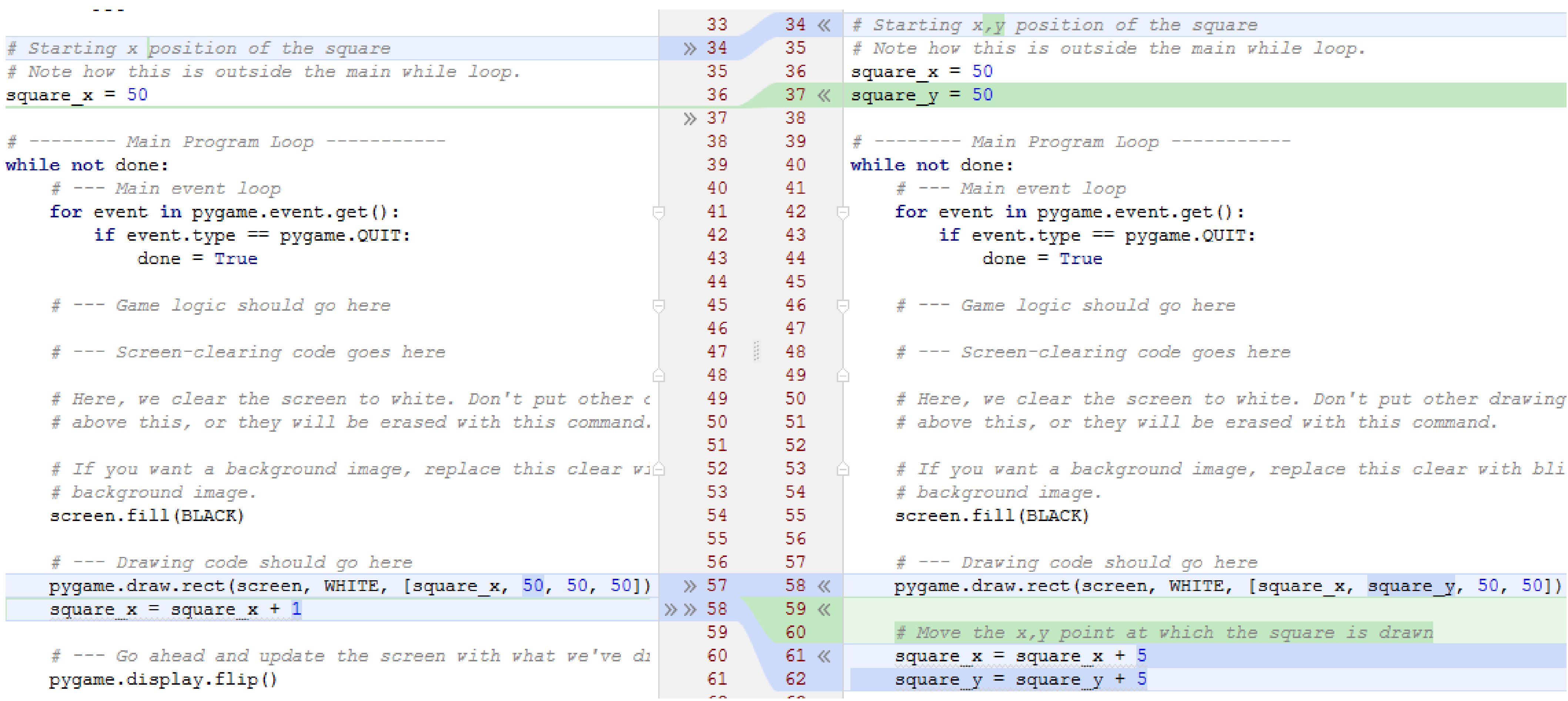
- Use 'square_y' when draw the 'square'



```
    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [square_x, square_y, 50, 50])

    # Move the x,y point at which the square is drawn
    square_x = square_x + 5
    square_y = square_y + 5
```

- Increase 'square_y' inside the loop by 1 slow, if increase by 5 goes faster

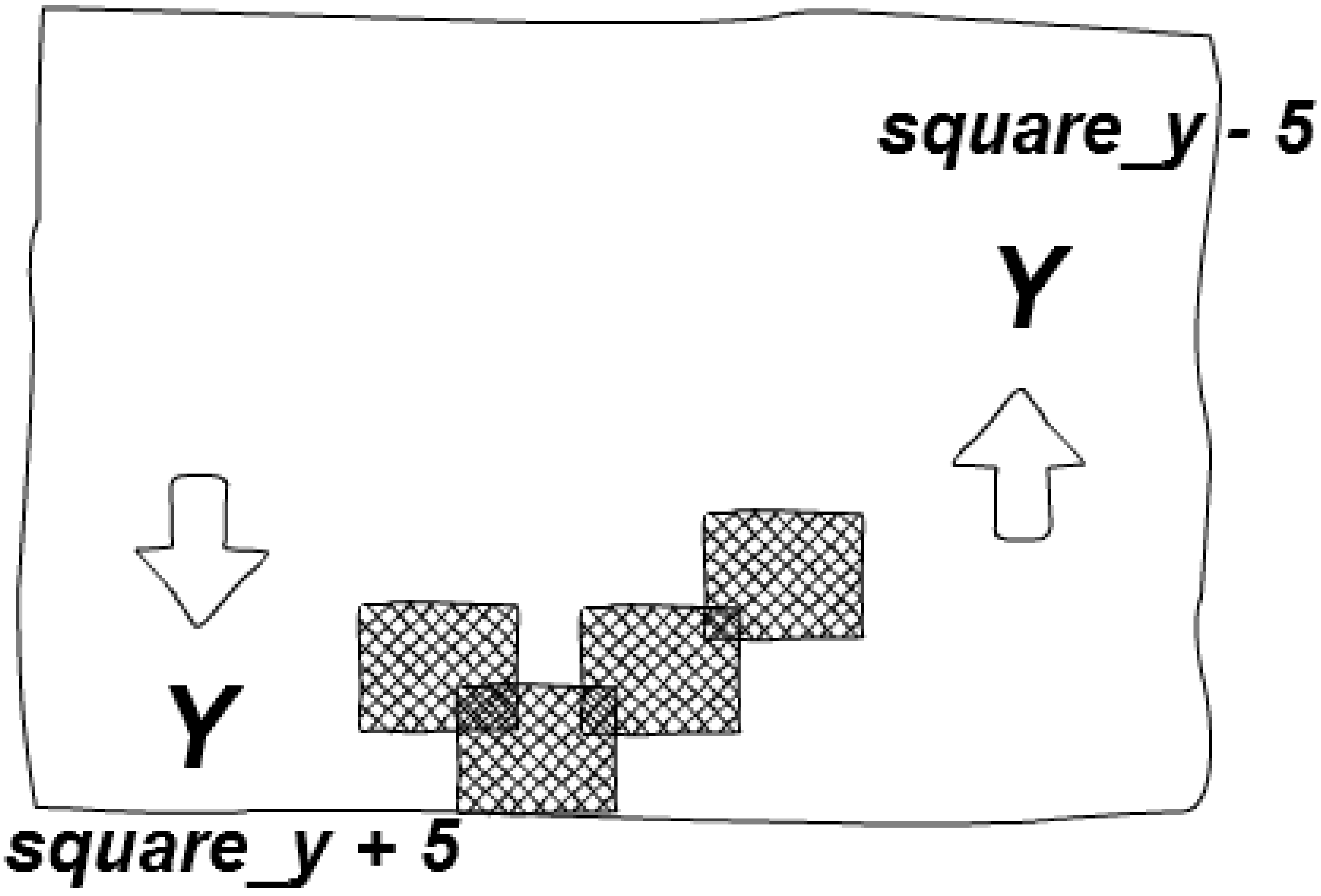
Move the square down & to the right



```
    ---
# Starting x position of the square
# Note how this is outside the main while loop.
square_x = 50

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # --- Game logic should go here

    # --- Screen-clearing code goes here

    # Here, we clear the screen to white. Don't put other c
    # above this, or they will be erased with this command.

    # If you want a background image, replace this clear wi
    # background image.
    screen.fill(BLACK)

    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [square_x, 50, 50, 50])
    square_x = square_x + 1

    # --- Go ahead and update the screen with what we've dı
    pygame.display.flip()
```

```
# Starting x,y position of the square
# Note how this is outside the main while loop.
square_x = 50
square_y = 50

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # --- Game logic should go here

    # --- Screen-clearing code goes here

    # Here, we clear the screen to white. Don't put other drawing
    # above this, or they will be erased with this command.

    # If you want a background image, replace this clear with bli
    # background image.
    screen.fill(BLACK)

    # --- Drawing code should go here
    pygame.draw.rect(screen, WHITE, [square_x, square_y, 50, 50])

    # Move the x,y point at which the square is drawn
    square_x = square_x + 5
    square_y = square_y + 5
```

Bounce off edges

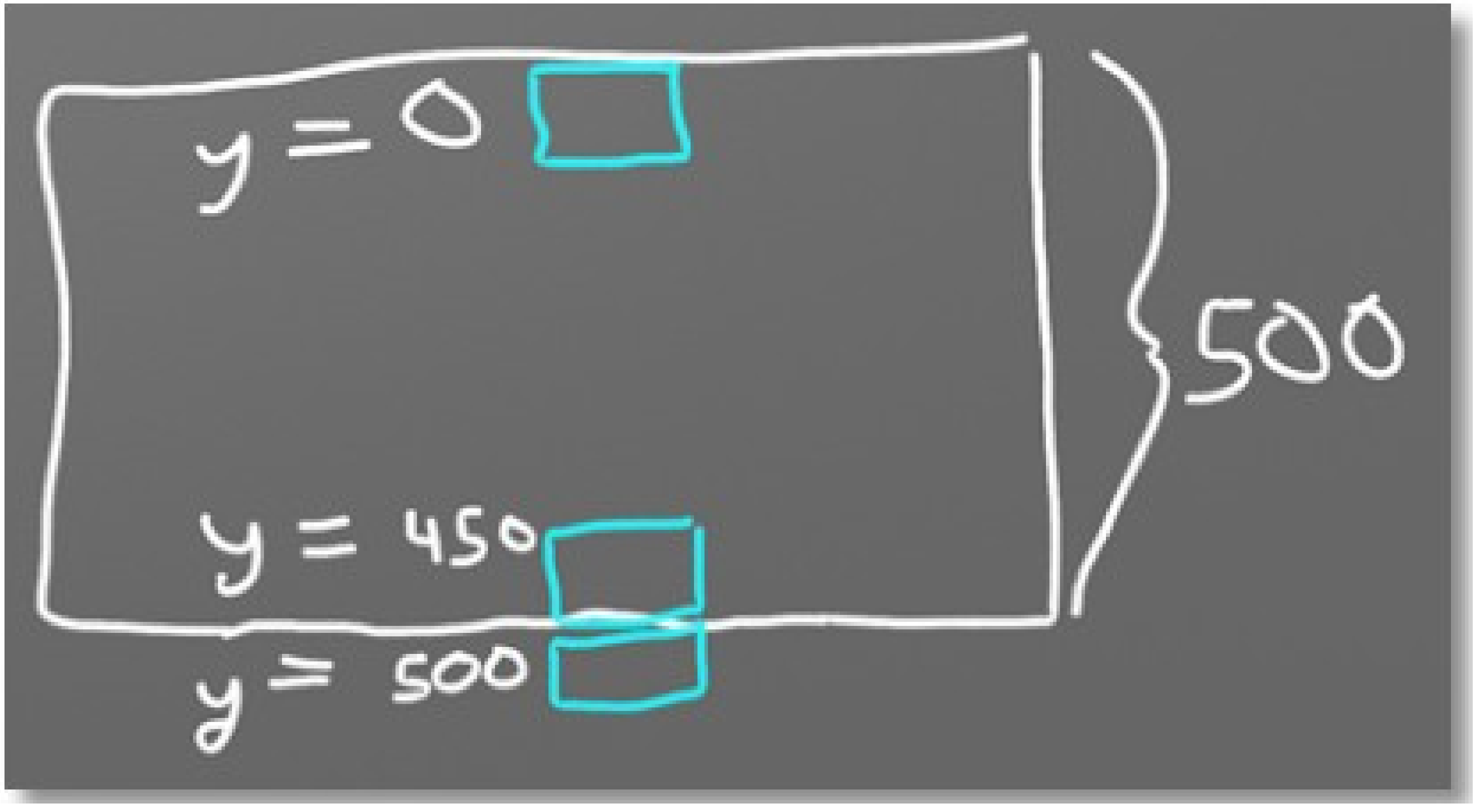


Bounce of edges

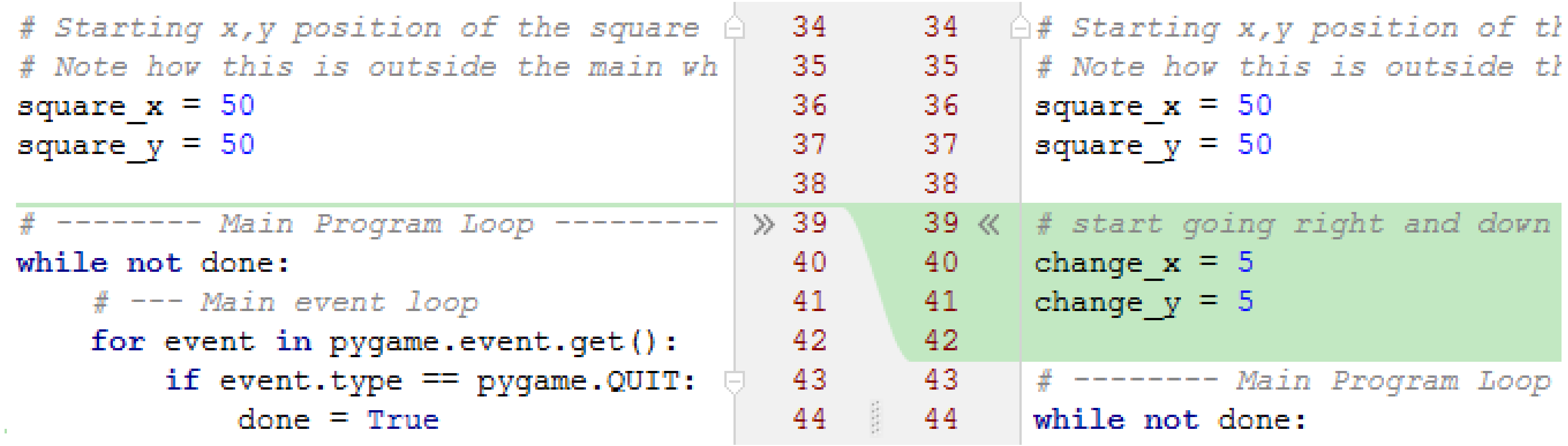
- “square_y = square_y + 5” goes down
- “square_y = square_y - 5” goes up
- So as hit edges need to reverse direction
- Use variable for amount of change 'change_y'
- “square_y = square_y + change_y”
- Start with “change_y = 5” going down
- As hit bottom edge “change_y = -5”
- As hit top edge “change_y = 5”

Bounce off edges

- When do you change direction?
  - At top where y = 0
  - At bottom where y = 450
  - Why 450 ?

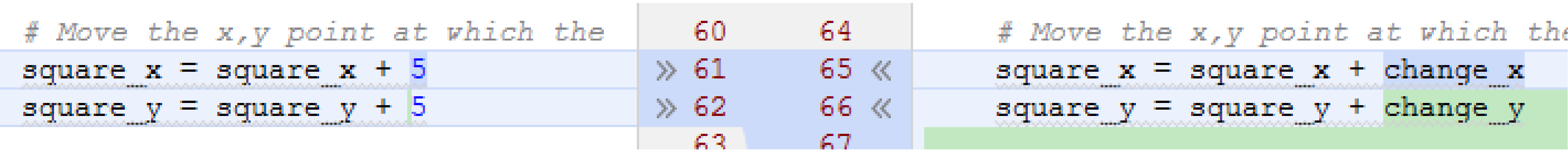


Bounce off edges – code changes



```
# Starting x,y position of the square
# Note how this is outside the main wh
square_x = 50
square_y = 50

# -------- Main Program Loop ---------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
```

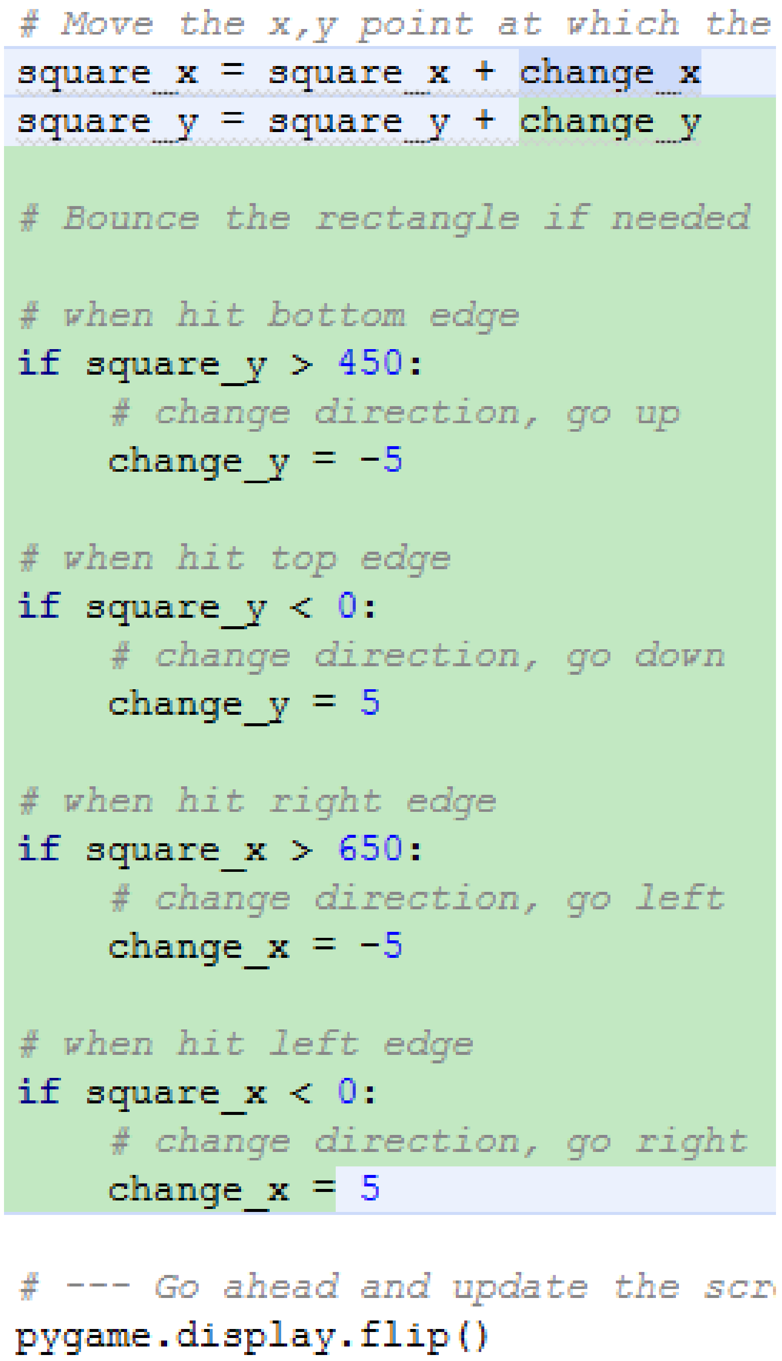
```
# Starting x,y position of th
# Note how this is outside th
square_x = 50
square_y = 50

# start going right and down
change_x = 5
change_y = 5

# -------- Main Program Loop
while not done:
```



```
# Move the x,y point at which the
square_x = square_x + 5
square_y = square_y + 5
```

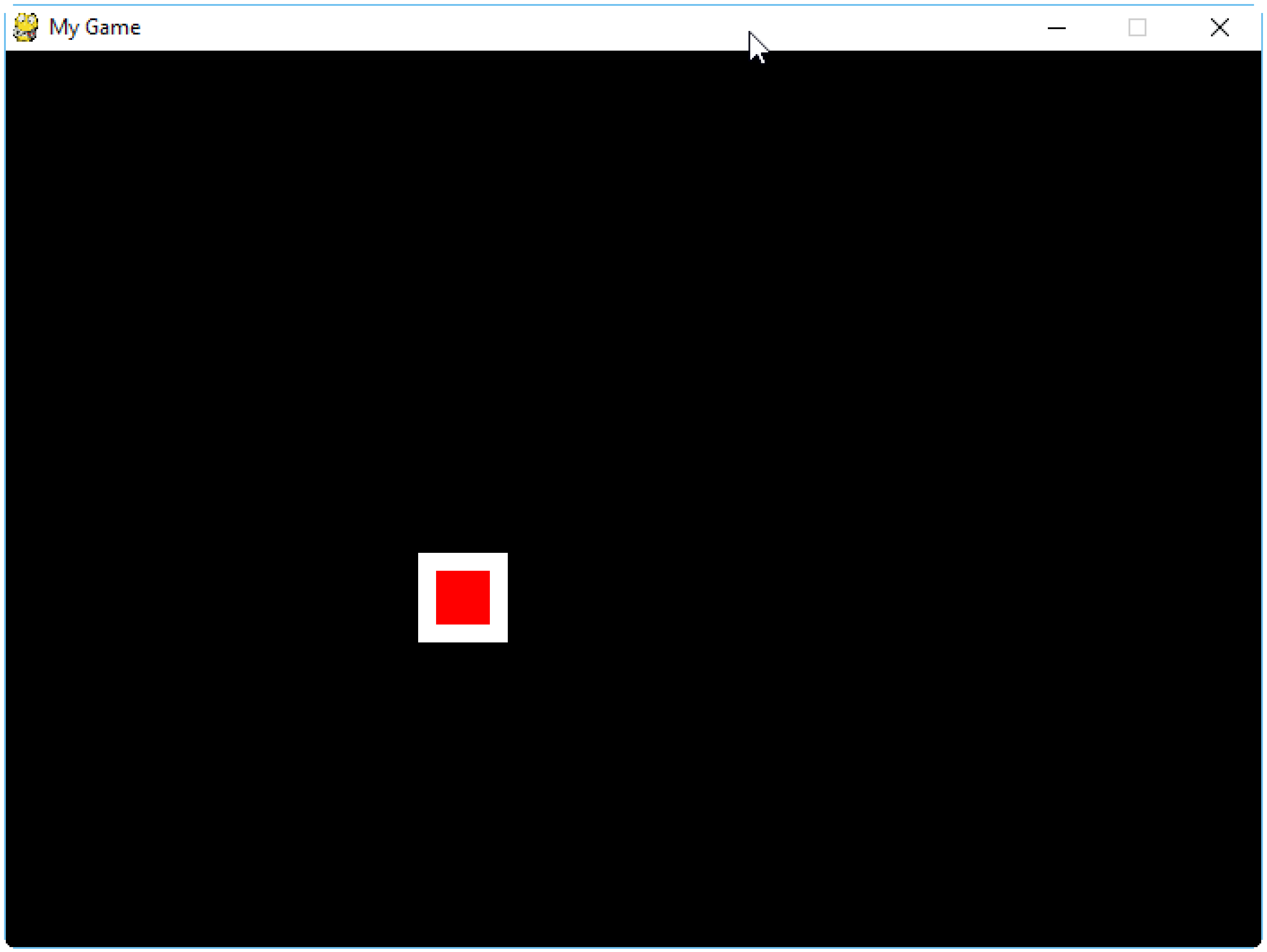
```
# Move the x,y point at which the
square_x = square_x + change_x
square_y = square_y + change_y
```

Bounce off edges – code changes



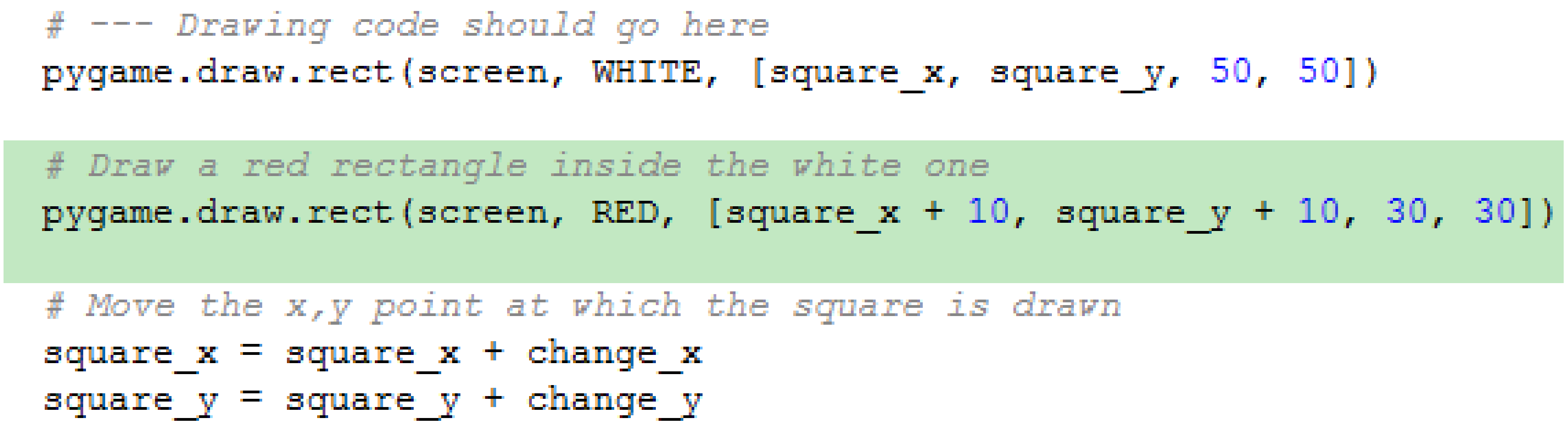
```
# Move the x,y point at which the
square_x = square_x + change_x
square_y = square_y + change_y

# Bounce the rectangle if needed

# when hit bottom edge
if square_y > 450:
    # change direction, go up
    change_y = -5

# when hit top edge
if square_y < 0:
    # change direction, go down
    change_y = 5

# when hit right edge
if square_x > 650:
    # change direction, go left
    change_x = -5

# when hit left edge
if square_x < 0:
    # change direction, go right
    change_x = 5

# --- Go ahead and update the scr
pygame.display.flip()
```

Red square inside white one



Red square inside... – code changes

- Draw red square offset by x +10 and y + 10
- Size 30 x 30



```
# --- Drawing code should go here
pygame.draw.rect(screen, WHITE, [square_x, square_y, 50, 50])

# Draw a red rectangle inside the white one
pygame.draw.rect(screen, RED, [square_x + 10, square_y + 10, 30, 30])

# Move the x,y point at which the square is drawn
square_x = square_x + change_x
square_y = square_y + change_y
```

Animating Snow

- We will change your copy of **snow.py**
- Very similar to 'bouncing square', 400 x 400
- 50 snow flakes start at random x, y position
- Flakes are small circles which fall, y = y + 1
- At the bottom of the screen, y > 400
  - Set 'y' to be < 0, off top of screen
  - Set 'x' to be random 0 .. 400
- Run it and see ...

Animating Snow - Changes

- Make snow flakes fall faster

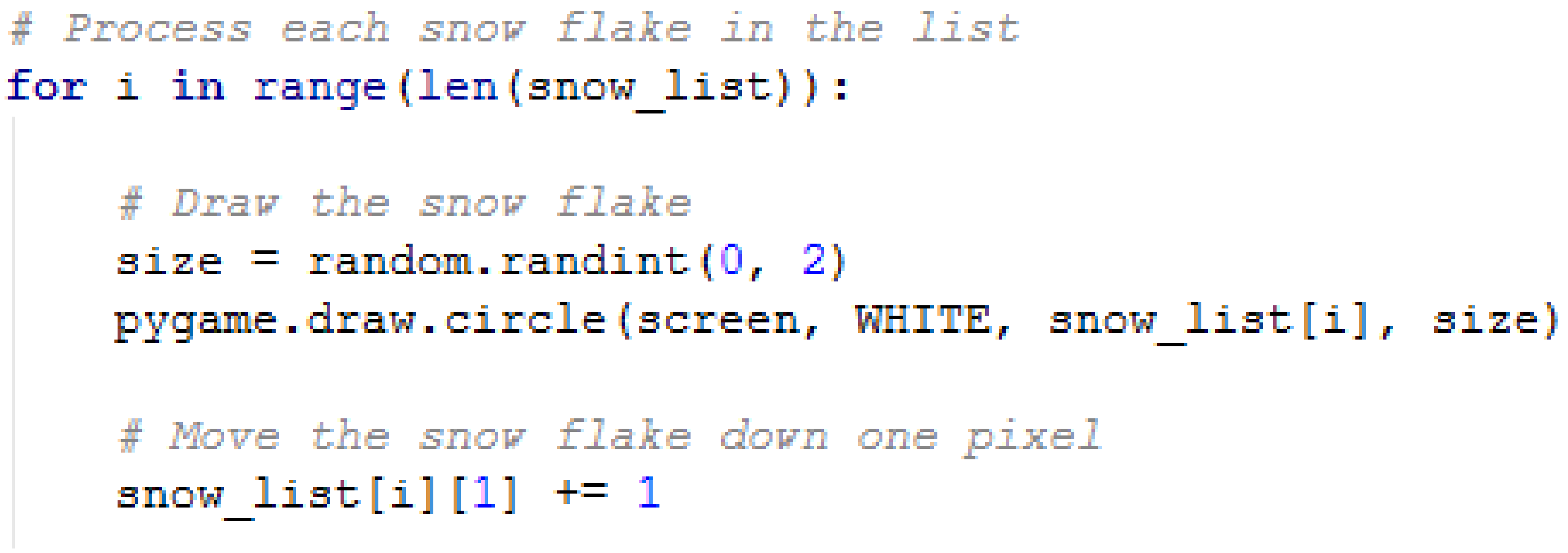
```
# Move the snow flake down one pixel
speed = 1
snow_list[i][1] += speed
```

- Make snow flakes bigger

```
# Draw the snow flake
size = 2
pygame.draw.circle(screen, snow_colours[i], snow_list[i], size)
```

Animating Snow - Changes

- Make snow flakes twinkle like stars



```
# Process each snow flake in the list
for i in range(len(snow_list)):

    # Draw the snow flake
    size = random.randint(0, 2)
    pygame.draw.circle(screen, WHITE, snow_list[i], size)

    # Move the snow flake down one pixel
    snow_list[i][1] += 1
```