CREATE A GAME OF ***BREAKOUT***</i></u> *USING **PYGAME***</i></u>

Step 1:** Import the module **"Pygame"

In [1]:
```
import pygame
```

```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

***Step 2:** Initialize all the modules and in return get the tuple of total execeptions*

In [2]:
```
pygame.init()
```

Out[2]:
```
(7, 0)
```

***Step 3:** Declare all the colors which we are going to use in **"Hex"** format*

In [3]:
```
WHITE = (255,255,255)
DARKBLUE = (36,90,190)
LIGHTBLUE = (0,176,240)
RED = (255,0,0)
```

***Step 4:** Using **"List Comprehension"** create a template for bricks by specifying the position parameter for 6 bricks in one line*

In [4]:
```
bricks1=[pygame.Rect(10 + i* 100,60,80,30) for i in range(6)]
bricks2=[pygame.Rect(10 + i* 100,100,80,30) for i in range(6)]
bricks3=[pygame.Rect(10 + i* 100,140,80,30) for i in range(6)]
```

***Step 5:** Create a **"Function"** to draw the bricks*

In [5]:
```
def draw_brick(brick_list):
    for i in brick_list:
        pygame.draw.rect(screen,RED,i)
```

***Step 6:** Initialize a **"Variable"** named **"Score"** to 0 to begin the score counter*

In [6]:
```
score = 0
```

***Step 7:** Create a **"List"** named **"Velocity"** to control the game physics dynamics*

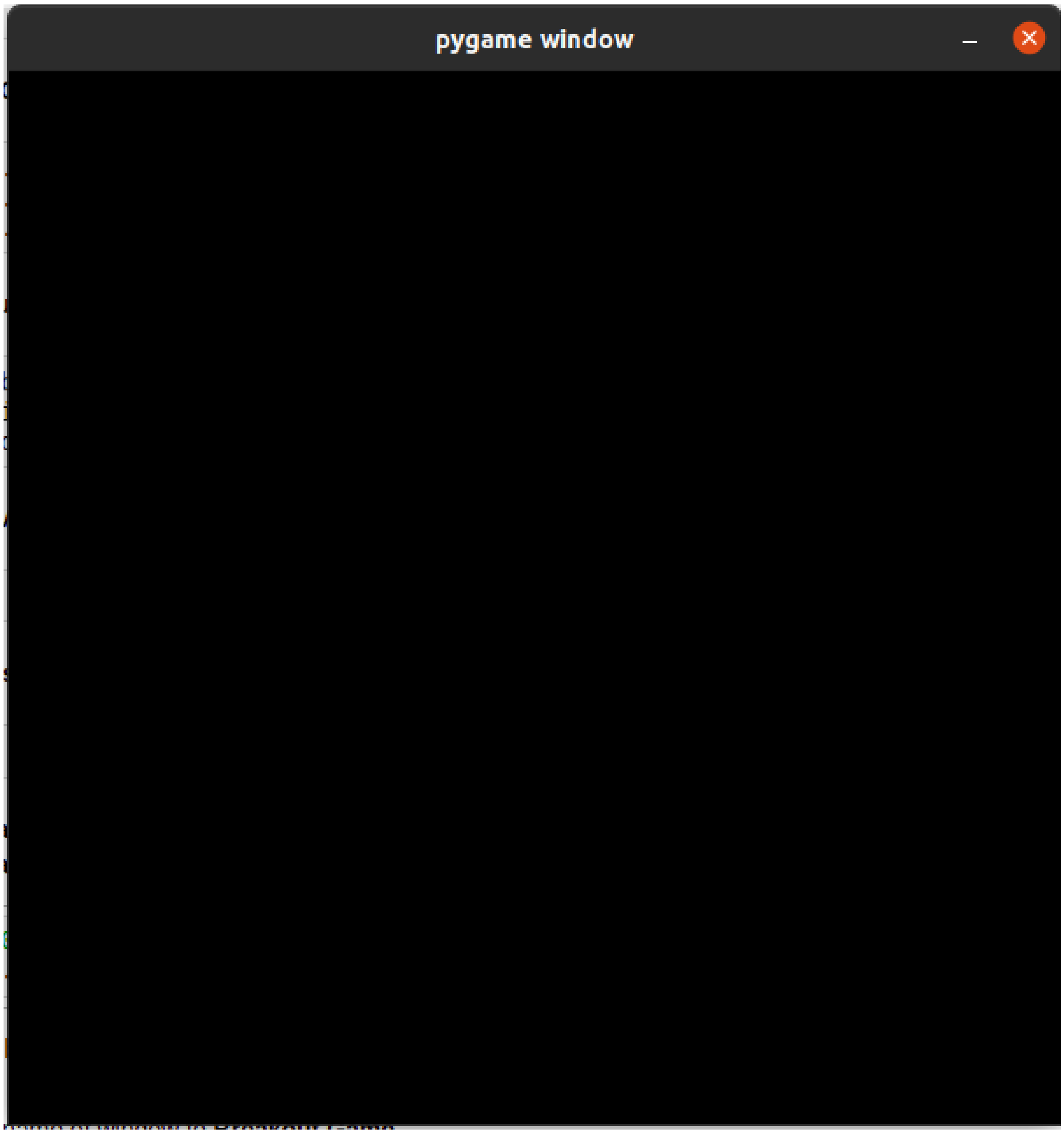
In [7]:
```
velocity=[1,1]
```

***Step 8:** Create a **"Variable"** named **"Size"** for storing the screen size and create another **"Variable"** to set the screen by passing the variable as a parameter in **"Display"** function in pygame*

In [8]:
```
size = (600, 600)
screen = pygame.display.set_mode(size)
```

(NOTE: If the above block of code if executed now will give you a plain black screen)

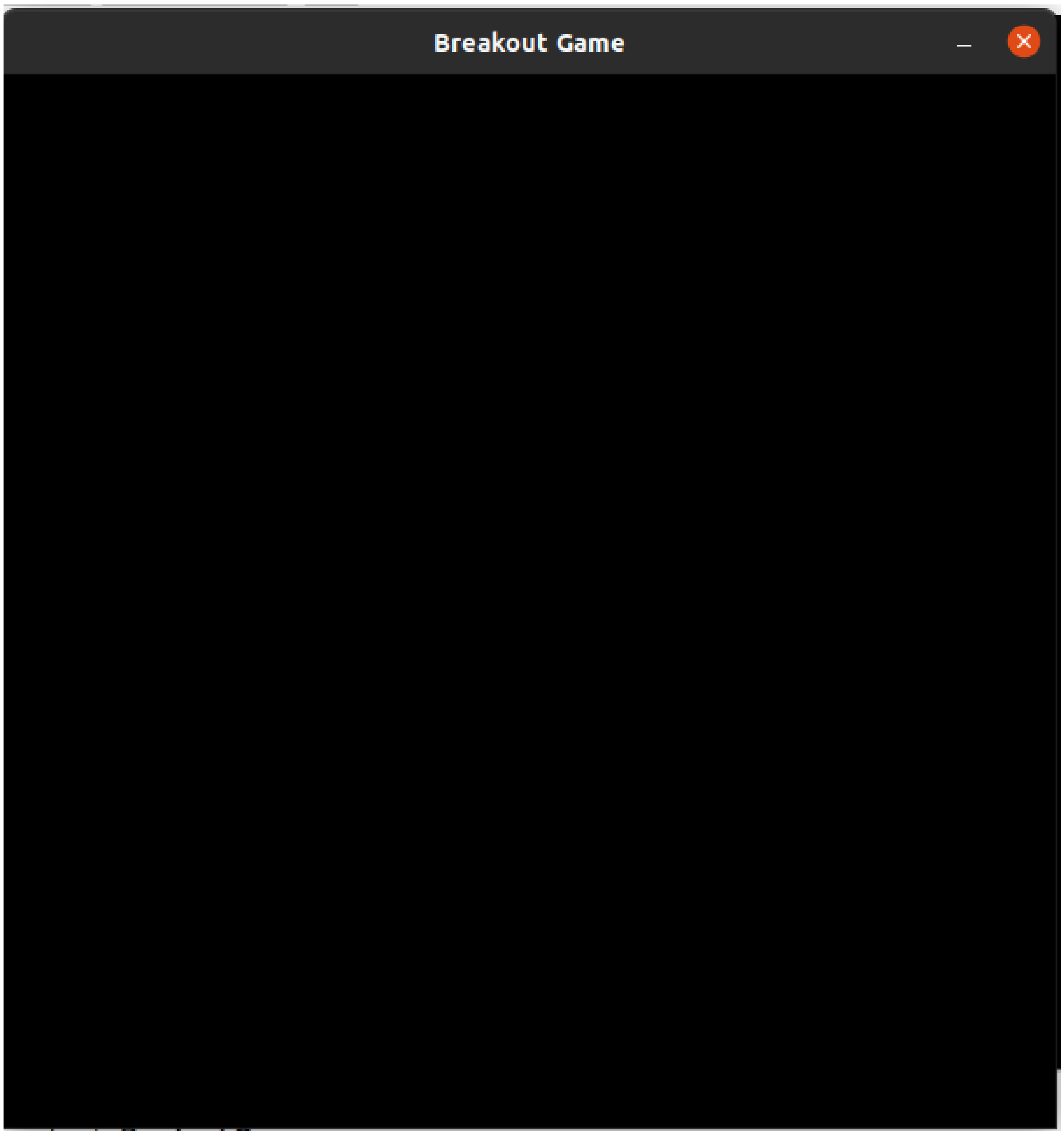
OUTPUT:



Step 9: *Change the name of window to* ***Breakout Game***

In [9]:
```python
pygame.display.set_caption("Breakout Game")
```

OUTPUT:



Step 10:** Create a template for **Paddle** and **Ball

In [10]:
```
paddle=pygame.Rect(300,550,60,10)
ball=pygame.Rect(200,250,10,10)
```

Step 11:** Create a flag variable named **carryOn** and set its value to **TRUE

In [11]:
```
carryOn = True
```

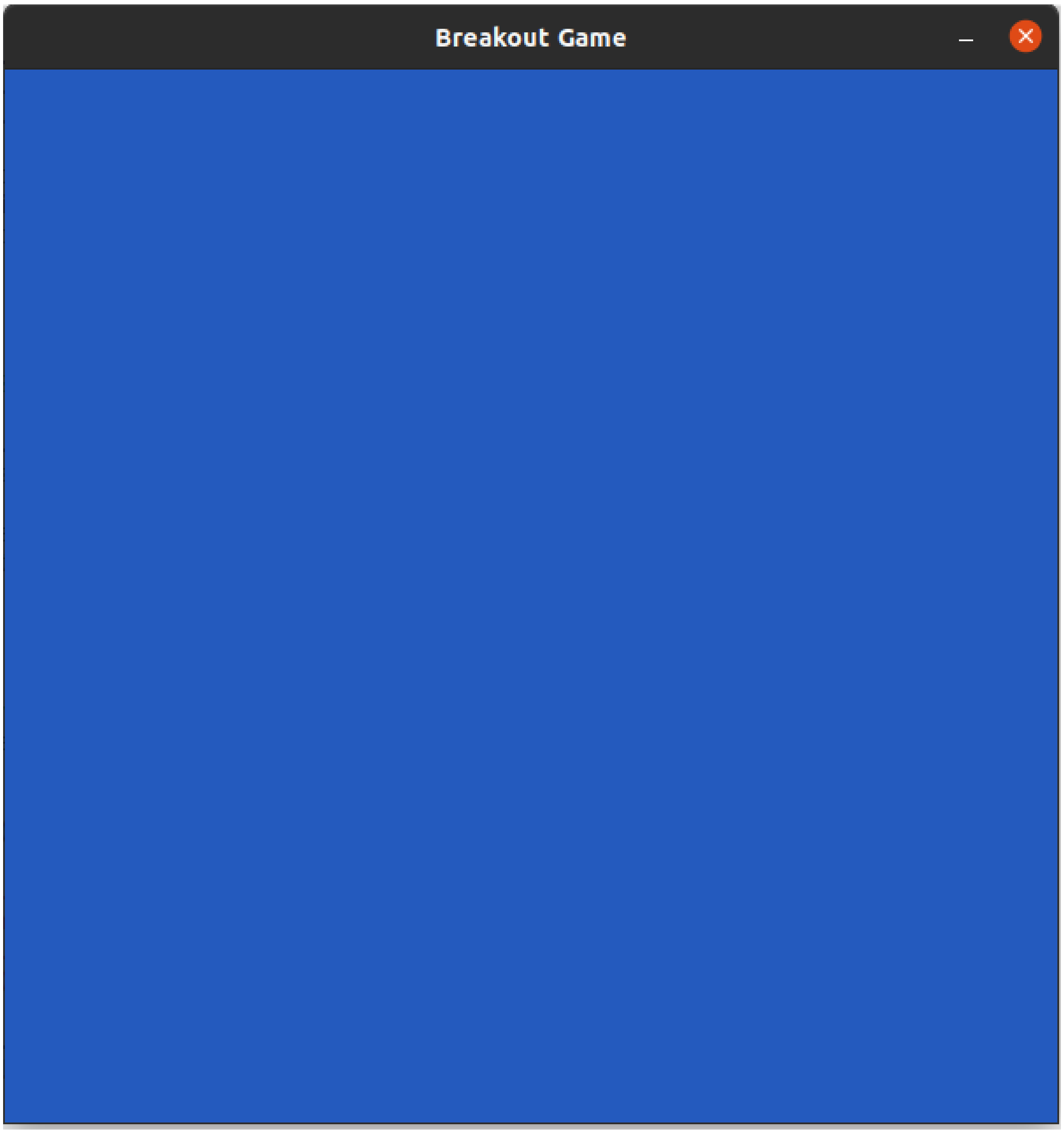
Step 12:** Create a **while loop** and use the flag variable so as to keep the loop running until **TRUE** and create a **for loop** to check whether user did something or not and Check whether user has pressed **QUIT** and if so set the flag to **FALSE** to break out of the **while loop

In [12]:
```
while carryOn:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            carryOn = False
```

***Step 13:** Fill the screen with **DARKBLUE** color*

In [13]:
```
     screen.fill(DARKBLUE)
```

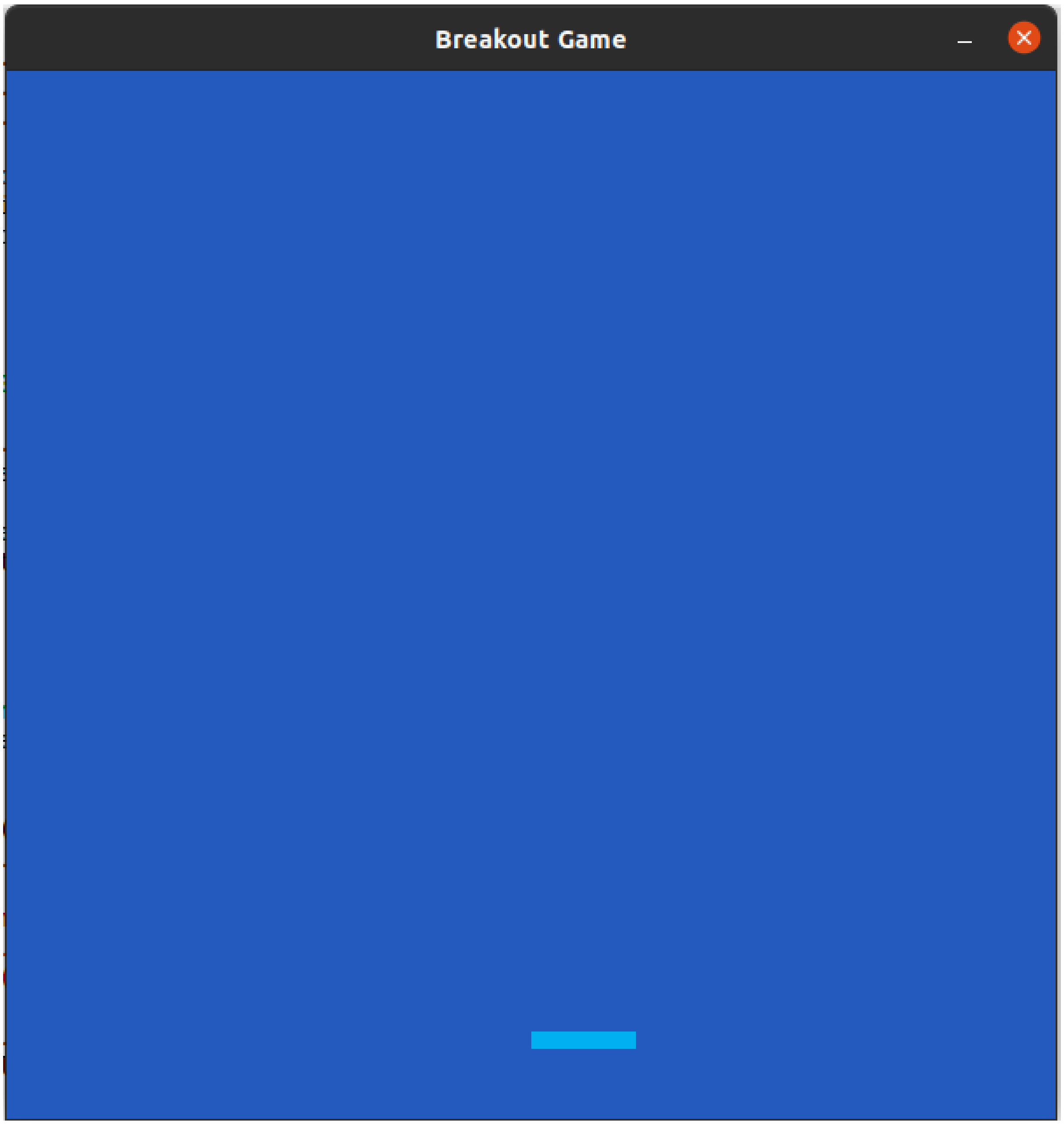
Out[13]: <rect(0, 0, 600, 600)>



Step 14: *A paddle is being drawn of* ***LIGHTBLUE*** *color*

In [14]:
```
    pygame.draw.rect(screen,LIGHTBLUE,paddle)
```

Out[14]: <rect(300, 550, 60, 10)>



In pygame text cannot be written directly to the screen.

Here, we have to print the score on the screen

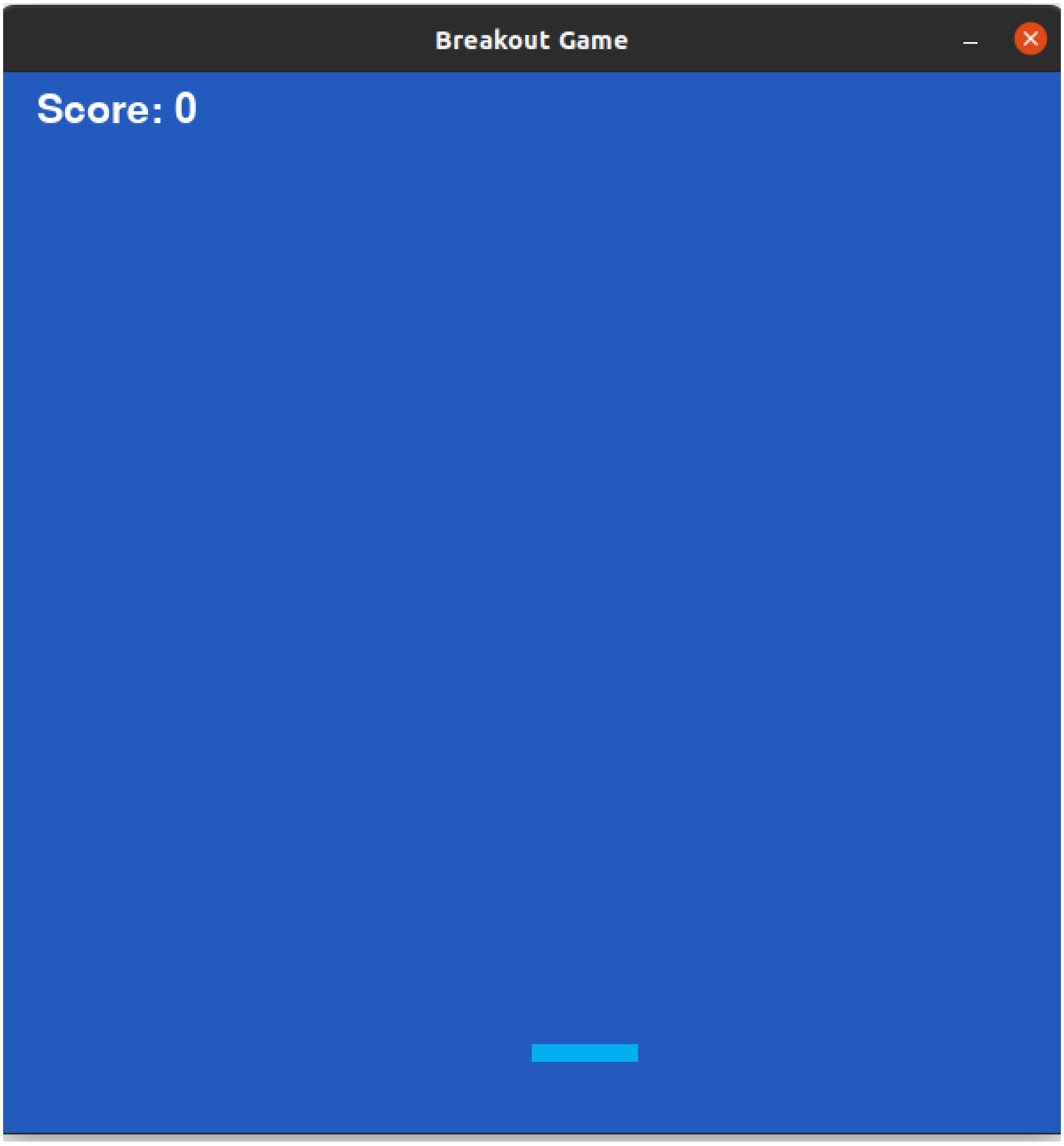
Step 15 *The first step is to create a Font object with given font size.*

Step 16 *The second step is to render the text into an image with a given color.*

Step 17 *The third step is to blit the image to the screen.*

In [15]:
```
    font = pygame.font.Font(None, 34)
    text = font.render("Score: " + str(score), 1, WHITE)
    screen.blit(text, (20,10))
```

Out[15]:
```
<rect(20, 10, 91, 23)>
```



Here we will code for the **Paddle movement**

Step 18: *The first step is to check whether any key is pressed*

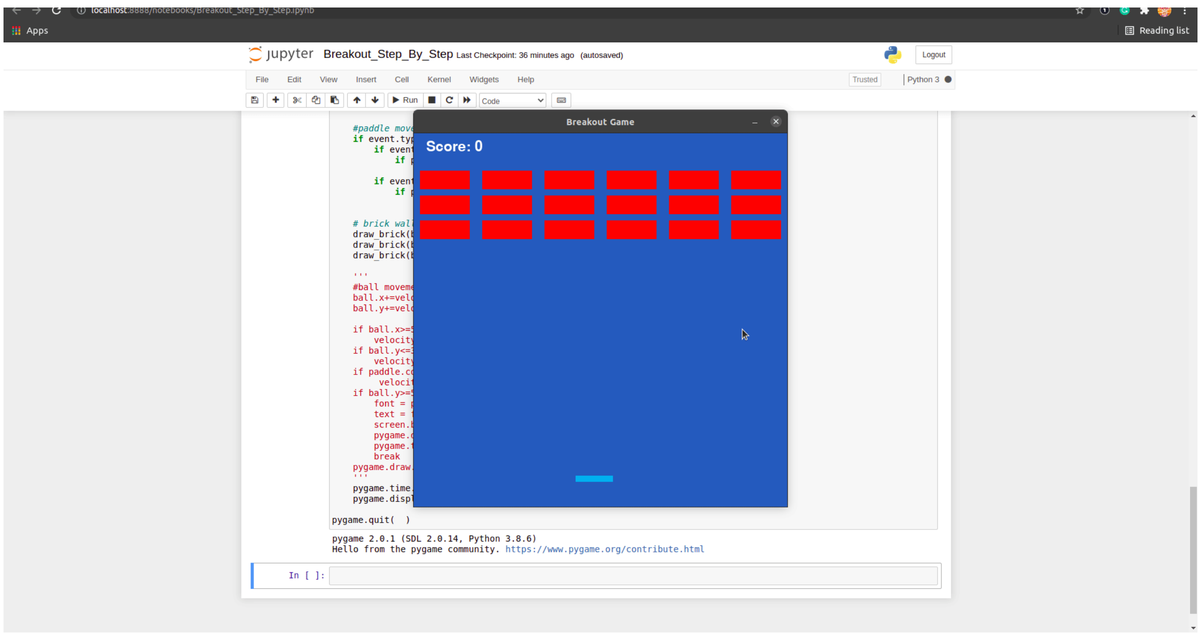
Step 19: *The second step is to check if key pressed is **Right** arrow key then we will check for the extreme right position i.e 540 using **If Block***

Step 20: *We have to move the paddle **5 pixels** towards right side so that the paddle could move inside of the screen*

Step 21: *The third step is to check if key pressed is **Left** arrow key then we will check for the extreme left position i.e 0 using **If Block***

Step 22: *We have to move the paddle **5 pixels** towards left side so that the paddle could move inside of the screen*

In [16]:
```
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RIGHT:
            if paddle.x<540:
                paddle.x+=5
        if event.key == pygame.K_LEFT:
            if paddle.x>0:
                paddle.x-=5
```



```
    #paddle move
    if event.typ
        if event
            if p

        if event
            if p


    # brick wal
    draw_brick(b
    draw_brick(b
    draw_brick(b

    '''
    #ball moveme
    ball.x+=velo
    ball.y+=velo

    if ball.x>=5
        velocity
    if ball.y<=3
        velocity
    if paddle.co
         velocit
    if ball.y>=5
        font = p
        text = f
        screen.b
        pygame.d
        pygame.t
        break
    pygame.draw.
    '''
    pygame.time.
    pygame.displ

pygame.quit(  )
```

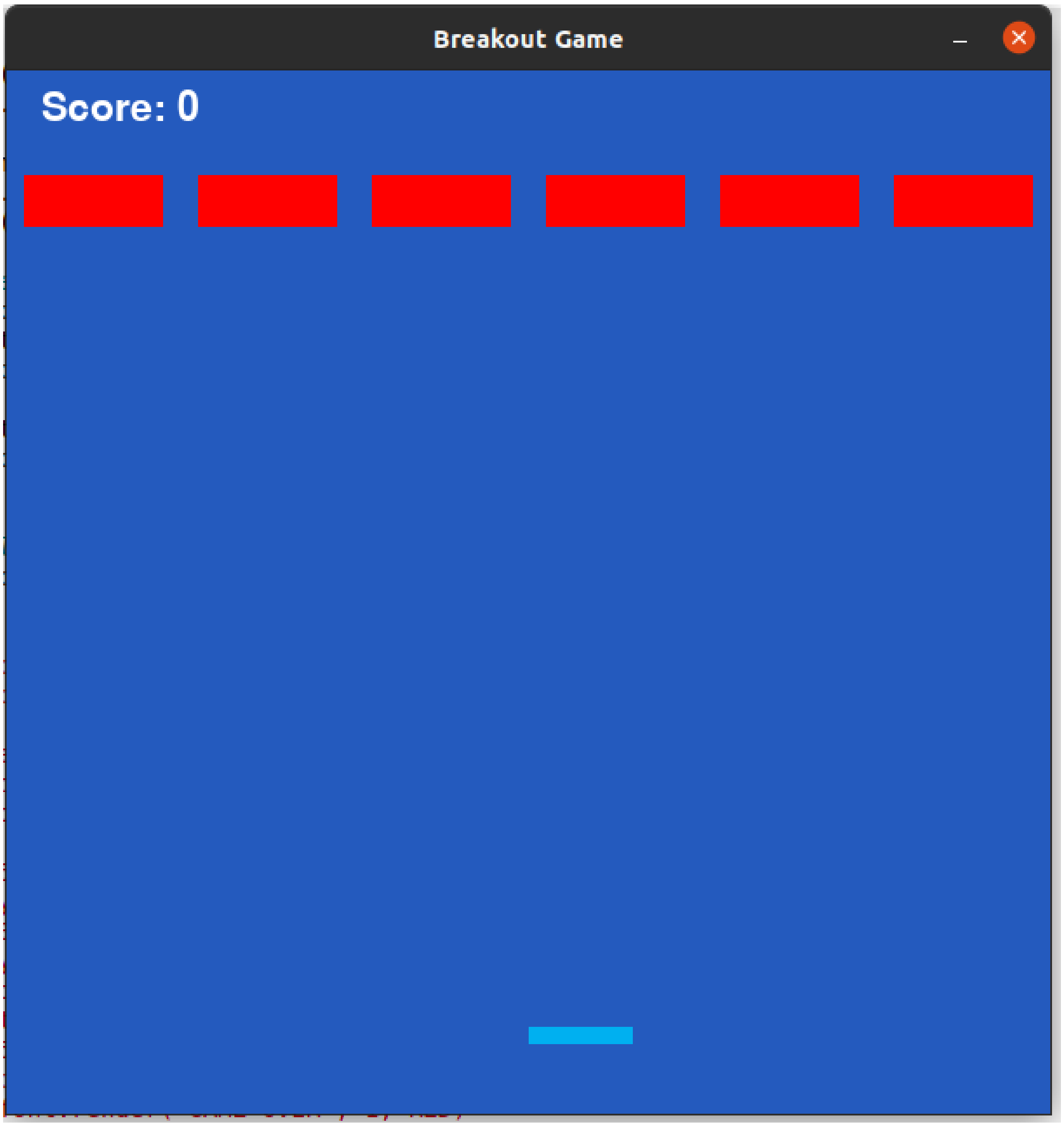
```
pygame 2.0.1 (SDL 2.0.14, Python 3.8.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

```
In [ ]:
```

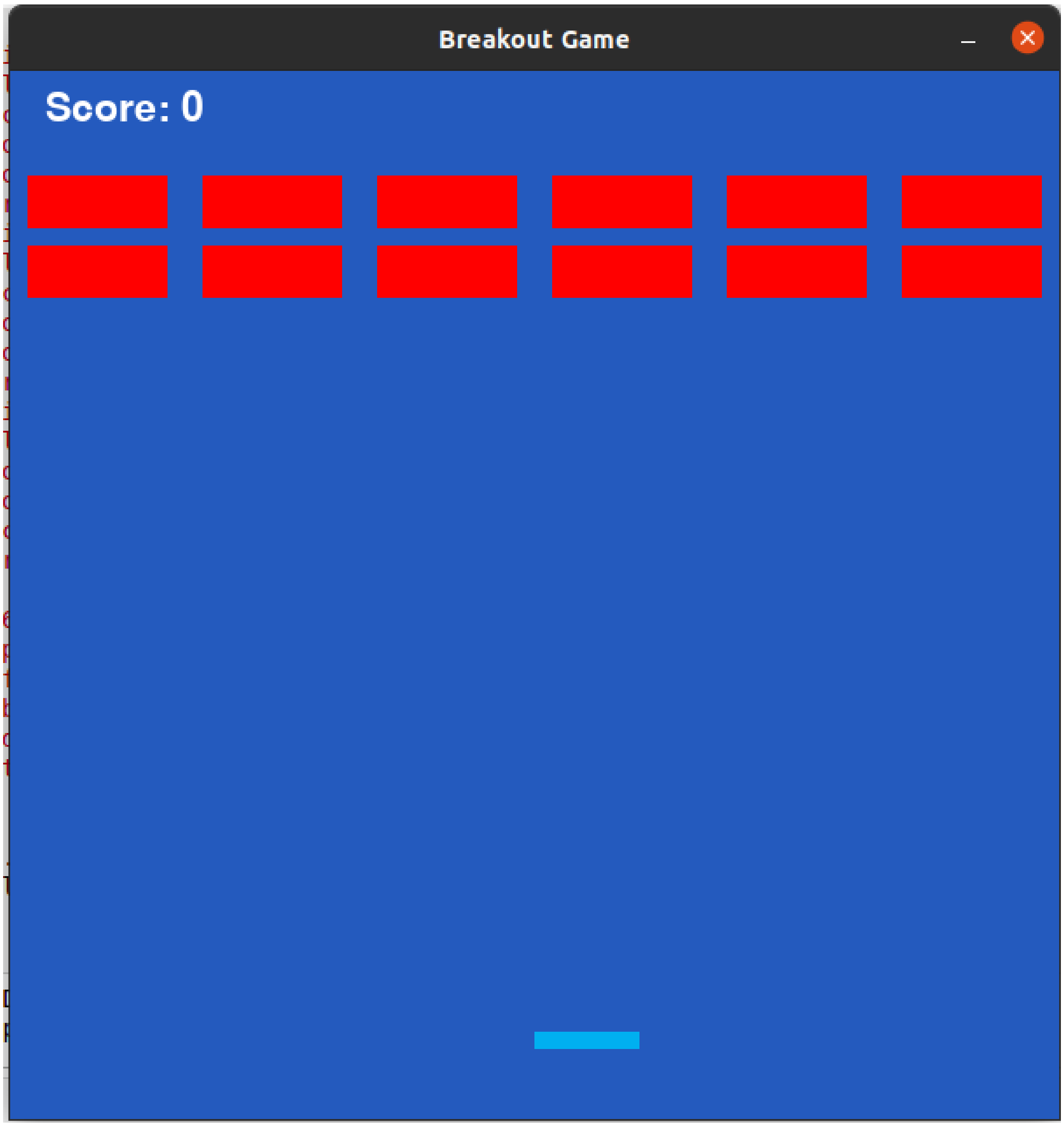
Step 23: *Call the function* ***draw_brick()*** *to draw the* ***Bricks***

In [17]:

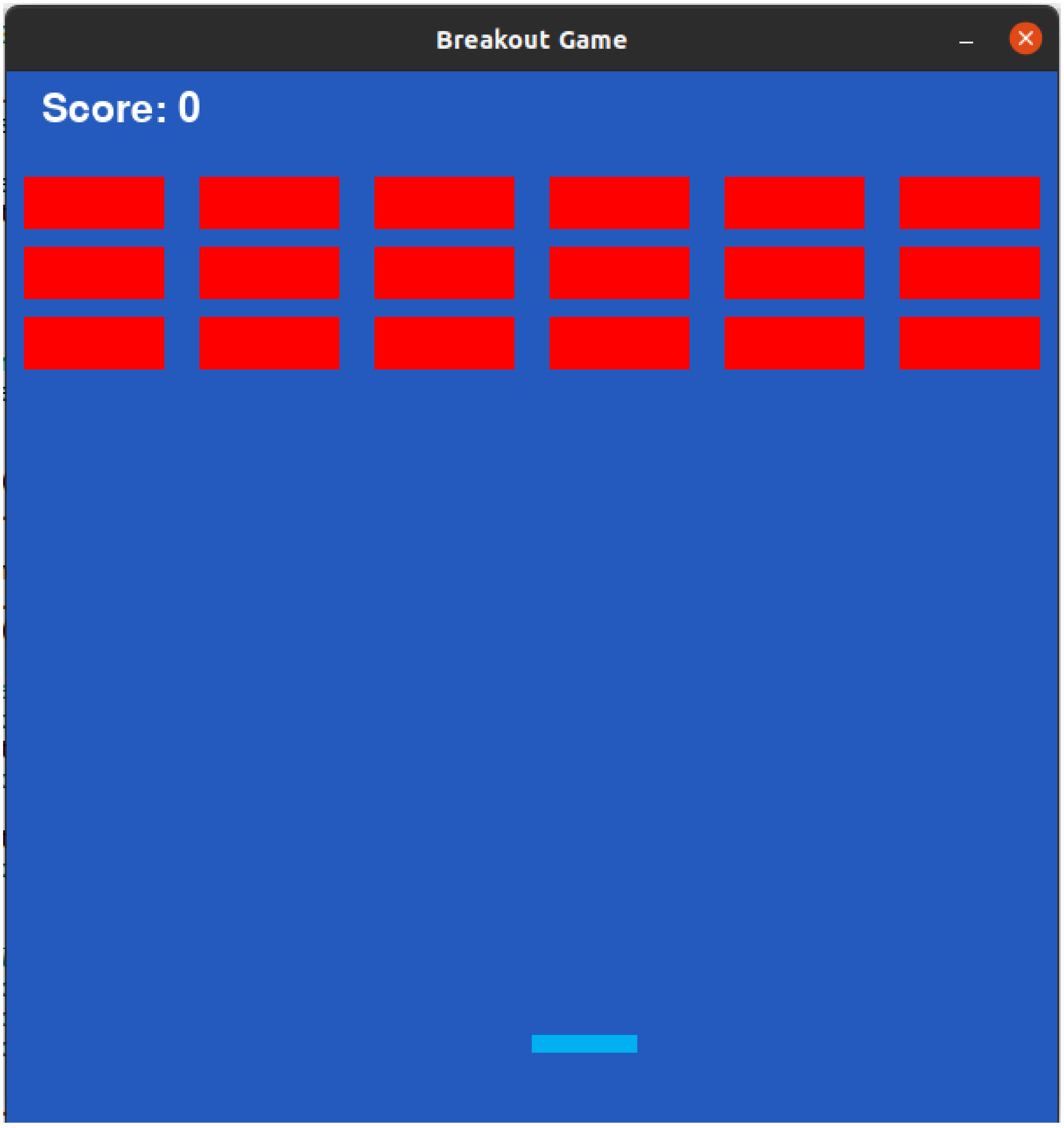
```
    draw_brick(bricks1)
```



In [ ]:
```
    draw_brick(bricks2)
```



In [ ]:
```
    draw_brick(bricks3)
```



***Step 24:** Change the ball's x and y **position**. Here we will make the ball move by incrementing using **velocity** and self*

In [18]:
```
    ball.x+=velocity[0]
    ball.y+=velocity[1]
```

Checking for ball and paddle maximum and minimum pixel position

***Step 25:** Check if the ball's x position has moved to extreme pixel values*

***Step 26:** Negate the velocity of ball so as it will maintain the physics dynamics of game as the ball has to return to the screen*

***Step 27:** Check if the ball's y position is less than 3 then negate the velocity of ball*

***Step 28:** Check if ball and paddle will collide then negate the velocity*

***Step 29:** Check if ball's y position touches the maximum pixel value*

In pygame text cannot be written directly to the screen.

Here, we have to print the GAME OVER message on the screen

    <b>Sub-Step 1</b> The first step is to create a Font object with given font size.

    <b>Sub-Step 2</b> The second step is to render the text into an image with a given color.

    <b>Sub-Step 3</b> The third step is to blit the image to the screen.

Step 30: *Now, here we will update the content on the screen*

Step 31: *We will provide a buffer for Game to end*

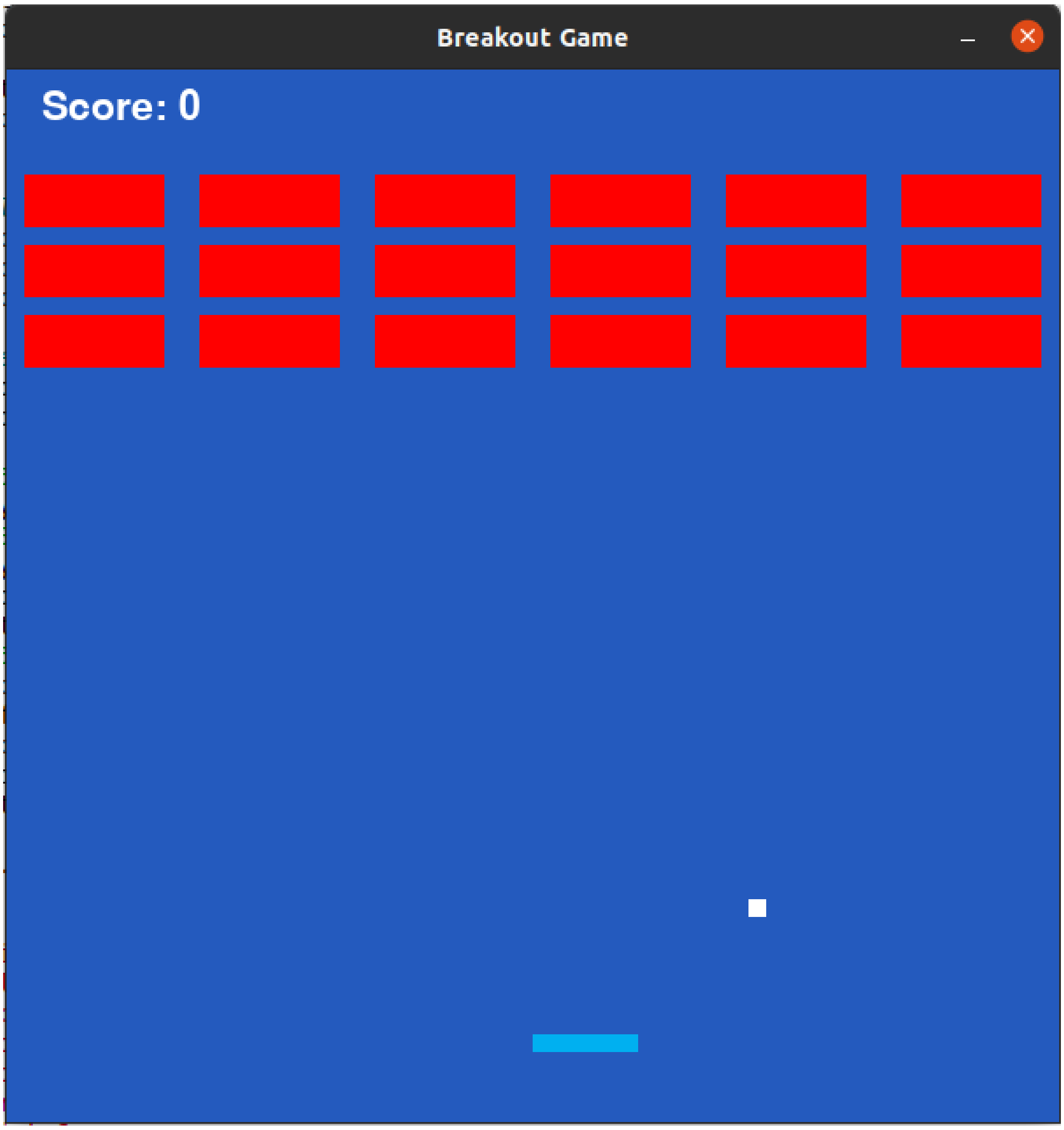
Step 32: *Use a **break** statement to exit the **nested-if loop***

Step 33: *Draw a ball of white color*

In [ ]:

```
    if ball.x>=590 or ball.x<=0:
        velocity[0] = -velocity[0]
    if ball.y<=3:
        velocity[1] = -velocity[1]
    if paddle.collidepoint(ball.x,ball.y):
        velocity[1]=-velocity[1]
    if ball.y>=590:
        font = pygame.font.Font(None, 74)
        text = font.render("GAME OVER", 1, RED)
        screen.blit(text, (150,350))
        pygame.display.flip()
        pygame.time.wait(2000)
        break

    pygame.draw.rect(screen,WHITE ,ball)
```



We will check for **Bricks** collision with **Ball**

Step 34: *Apply the **for loop** on **bricks1** so that we could check which brick in **bricks1** is being hit by ball*

Step 35: *Check if the **brick** and **ball** have collided using **if loop***

Step 36: *Remove the brick by using **list.remove()** built-in function*

Step 37: *Negate both the **velocity**list elements to maintain the game dynamics*

Step 38: *Increase the score by 1*

Step 39: *Similarly code for **bricks2** and **bricks3***

In [ ]:
```
    for i in bricks1:
        if i.collidepoint(ball.x,ball.y):
            bricks1.remove(i)
            velocity[0] = -velocity[0]
            velocity[1]=-velocity[1]
```

```
                score+=1
            
        for i in bricks2:
            if i.collidepoint(ball.x,ball.y):
                bricks2.remove(i)
                velocity[0] = -velocity[0]
                velocity[1]=-velocity[1]
                score+=1
            
        for i in bricks3:
            if i.collidepoint(ball.x,ball.y):
                bricks3.remove(i)
                velocity[0] = -velocity[0]
                velocity[1]=-velocity[1]
                score+=1 
```

Check whether maximum score is acheived while playing

In pygame text cannot be written directly to the screen.

Here, we have to print the YOU WON message on the screen

```
<b>Sub-Step 1</b> The first step is to create a Font object with
given font size. 

<b>Sub-Step 2</b> The second step is to render the text into an 
image with a given color. 

<b>Sub-Step 3</b> The third step is to blit the image to the 
screen.
```

Step 40: *Now, here we will update the content on the screen*

Step 41: *We will provide a buffer for Game to end*

Step 42: *Exit the loop if condition is satisfied*

In [ ]:
```
    if score==18:
        font = pygame.font.Font(None, 74)
        text = font.render("YOU WON!!", 1, RED)
        screen.blit(text, (150,350))
        pygame.display.flip()
        pygame.time.wait(2000)
        break
```

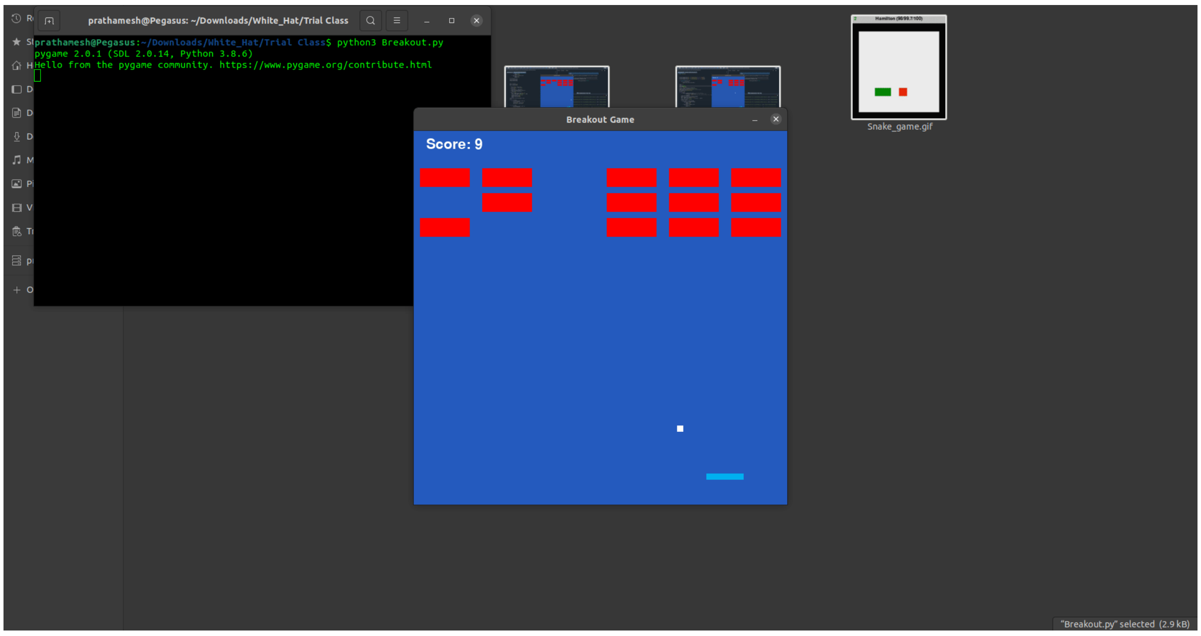
Step 43: *We will provide a buffer for Game to end*

Step 44: *Update the display on the screen*

In [ ]:
```
    pygame.time.wait(1)
    pygame.display.flip()
```

In [ ]:
```
<b>Step 45:</b> We will provide a buffer for Game to end
```

In [ ]:
```
pygame.quit(  )
```



```
prathamesh@Pegasus:~/Downloads/White_Hat/Trial Class$ python3 Breakout.py
pygame 2.0.1 (SDL 2.0.14, Python 3.8.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

prathamesh@Pegasus: ~/Downloads/White_Hat/Trial Class
Breakout Game
Score: 9
Snake_game.gif
"Breakout.py" selected (2.9 kB)