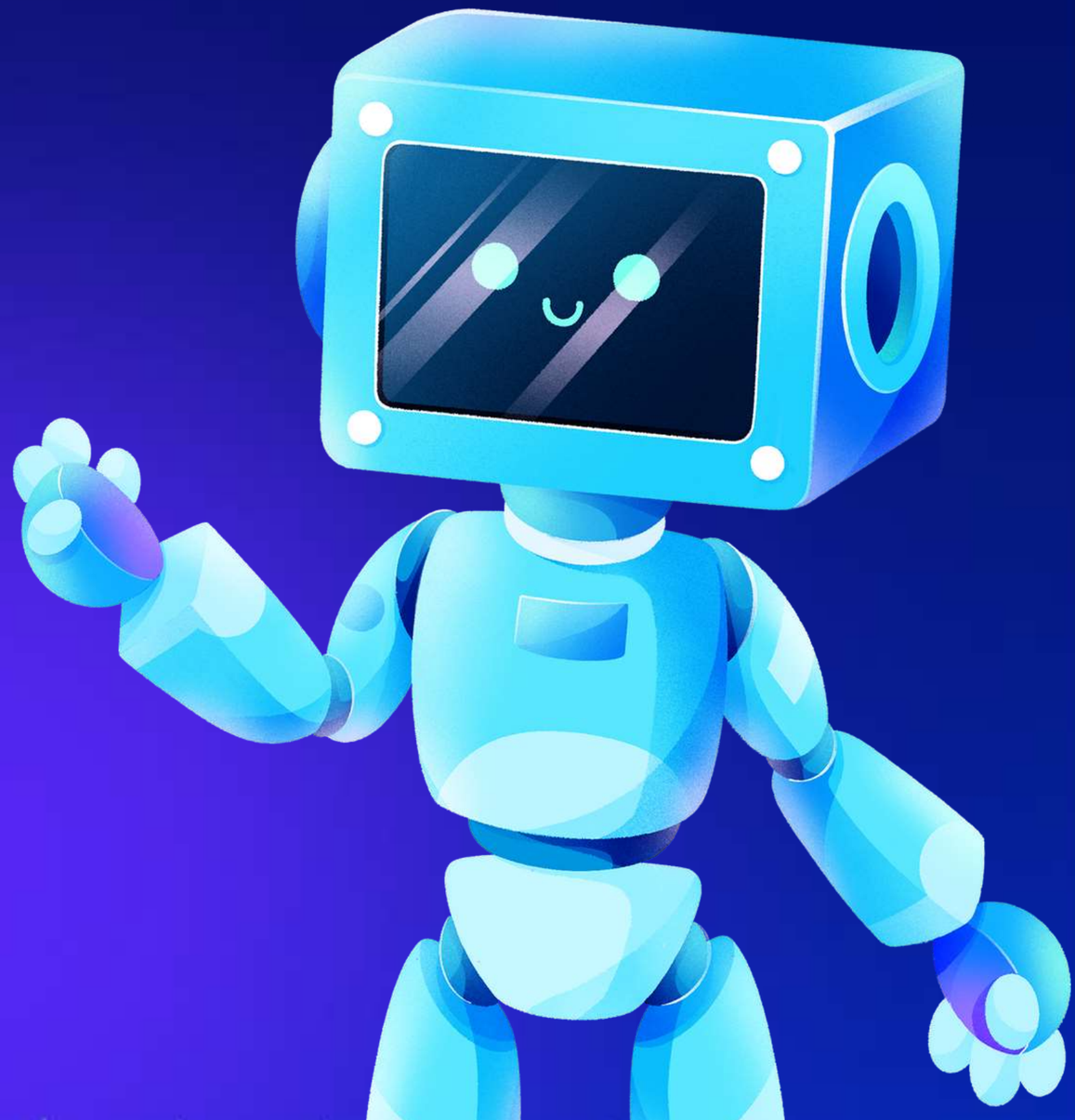
GAME DEVELOPMENT USING

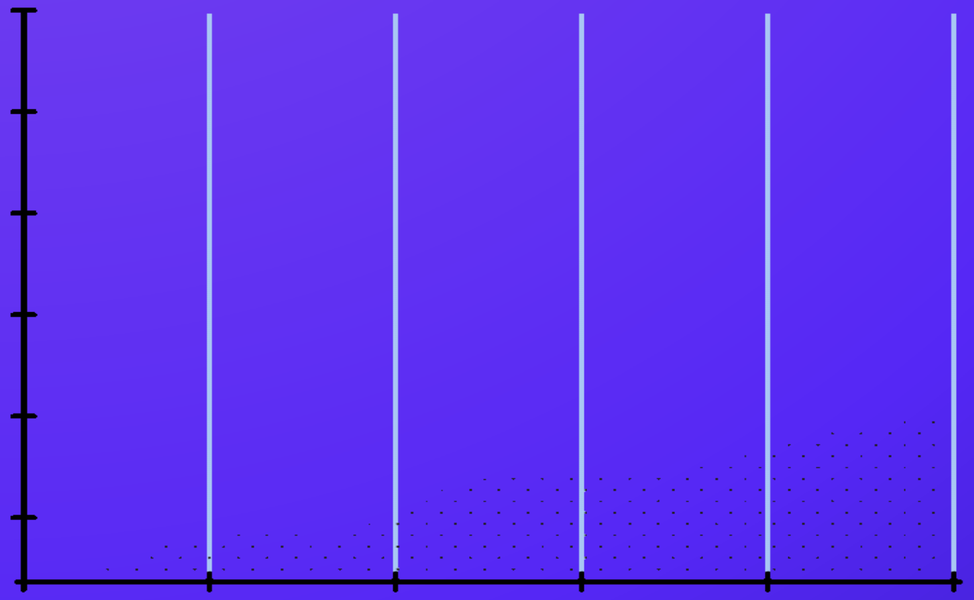
PYGAME

By MANOGNA, SATHVIKA and NITYA

MENTOR - SANVIT KATREKAR



WHAT WE LEARNT



DAY ONE:

Pygame first session

Installation

1. Installation of python - Go to website, get installer

- Problem:

  'pip' not found

- Solution: Go to python installer > Modify > Add python to path

2. install pygame:

   pip install pygame

3. Initialising pygame:

```
import pygame
pygame.init()
```

Basic Coding

1. Creating a window

```
window = pygame.display.set_mode((500,500))
#set_mode((screen width,screen height))
pygame.display.set_caption("first game")
x = 250
y = 250
width = 50
width = 50
velocity = 5 #speed with which the block moves
```

2. Main loop (while)

```
run = True
while run:
    pygame.time.delay(20) #so that the block doesn't move fast
```

3. Checking for event

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        run = False
#when close is clicked the code stops running and doesn't give an error

.
.
.
pygame.quit()
```

> x, y, width, height

- assign constants instead of writing value directly

  eg: delay_in_ms (units)

5. Moving the rectangle

```
keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT]:
    x -= velocity
if keys[pygame.K_RIGHT]:
    x += velocity
if keys[pygame.K_UP]:
    y -= velocity
if keys[pygame.K_DOWN]:
    y += velocity

window.fill("black")    #so that the block doesn't leave behind a trail
```

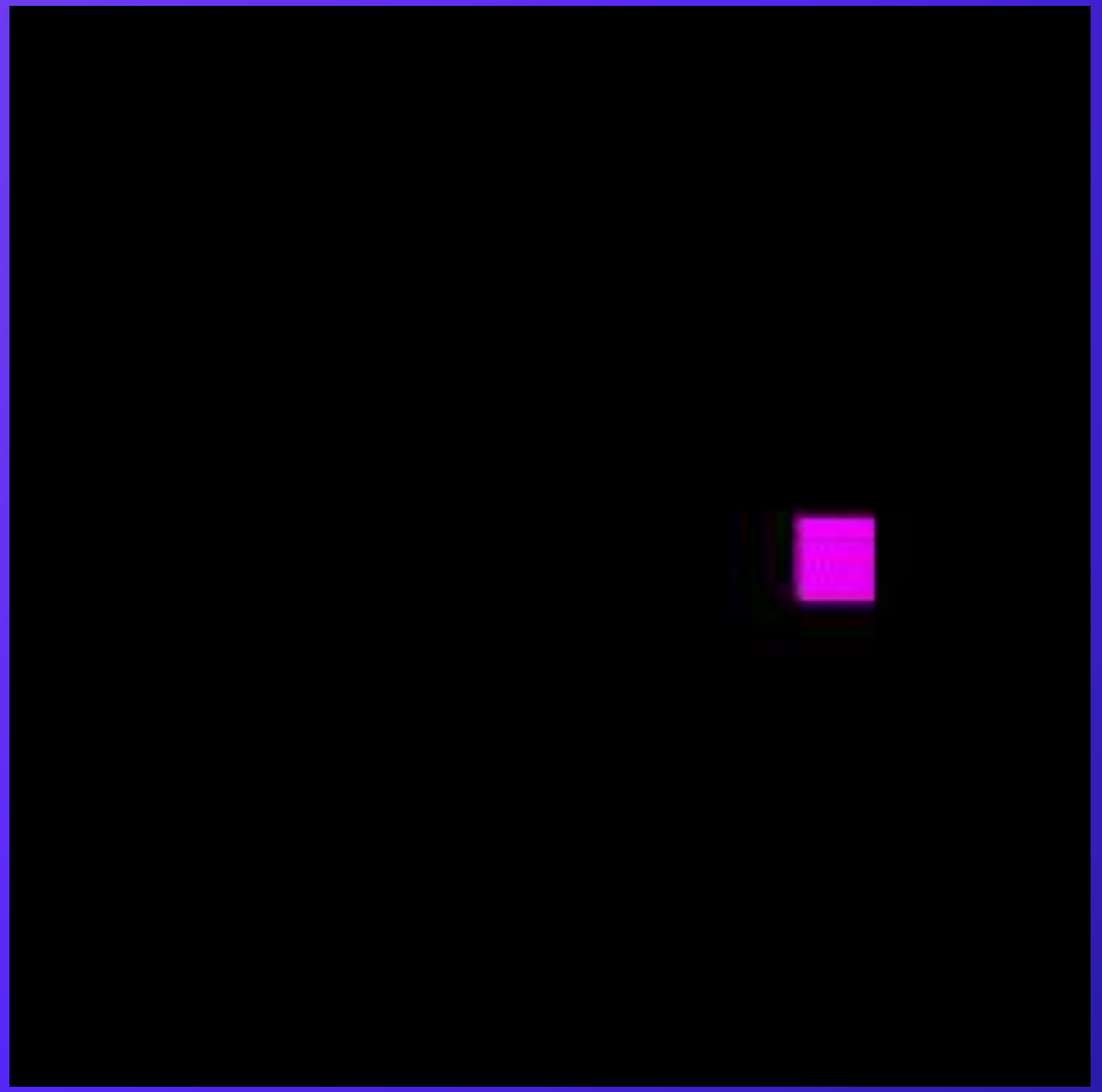
4. Creating the rectangle

```
pygame.draw.rect(window,(255,0,0),(x,y,width,height))
#(screen, color of rectangle, dimensions)
pygame.display.update() #makes the rectangle appear
```

> Colour of the rectangle

- RGB (Red, Green, Blue)
- 2^8: maximum value
- 255: range
- 0: no colour

SOME MORE...



DAY TWO:

Pygame Second Session

Using WASD Keys for Movement

```
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_w:
        up = True
        left, right, down = False, False, False
    if event.key == pygame.K_s:
        down = True
        left, right, up = False, False, False
    if event.key == pygame.K_a:
        left = True
        up, down, right = False, False, False
    if event.key == pygame.K_d:
        right = True
        left, up, down = False, False, False
```

Setting boundaries for the Rectangle

Problem: The rectangle moves out of the screen and doesn't come back

Solution: Setting boundaries so that it stops when it reaches one end.

```
if keys[pygame.K_LEFT] and x > velocity:
    x -= velocity
if keys[pygame.K_RIGHT] and x < SCREENX - width - velocity:
    x += velocity
if keys[pygame.K_UP] and y > velocity:
    y -= velocity
if keys[pygame.K_DOWN] and y < SCREENY - height - velocity:
    y += velocity
```

Defining left, right, up and down

```
if left:
    x -=vel
if right:
    x += vel
if up:
    y -= vel
if down:
    y += vel
```

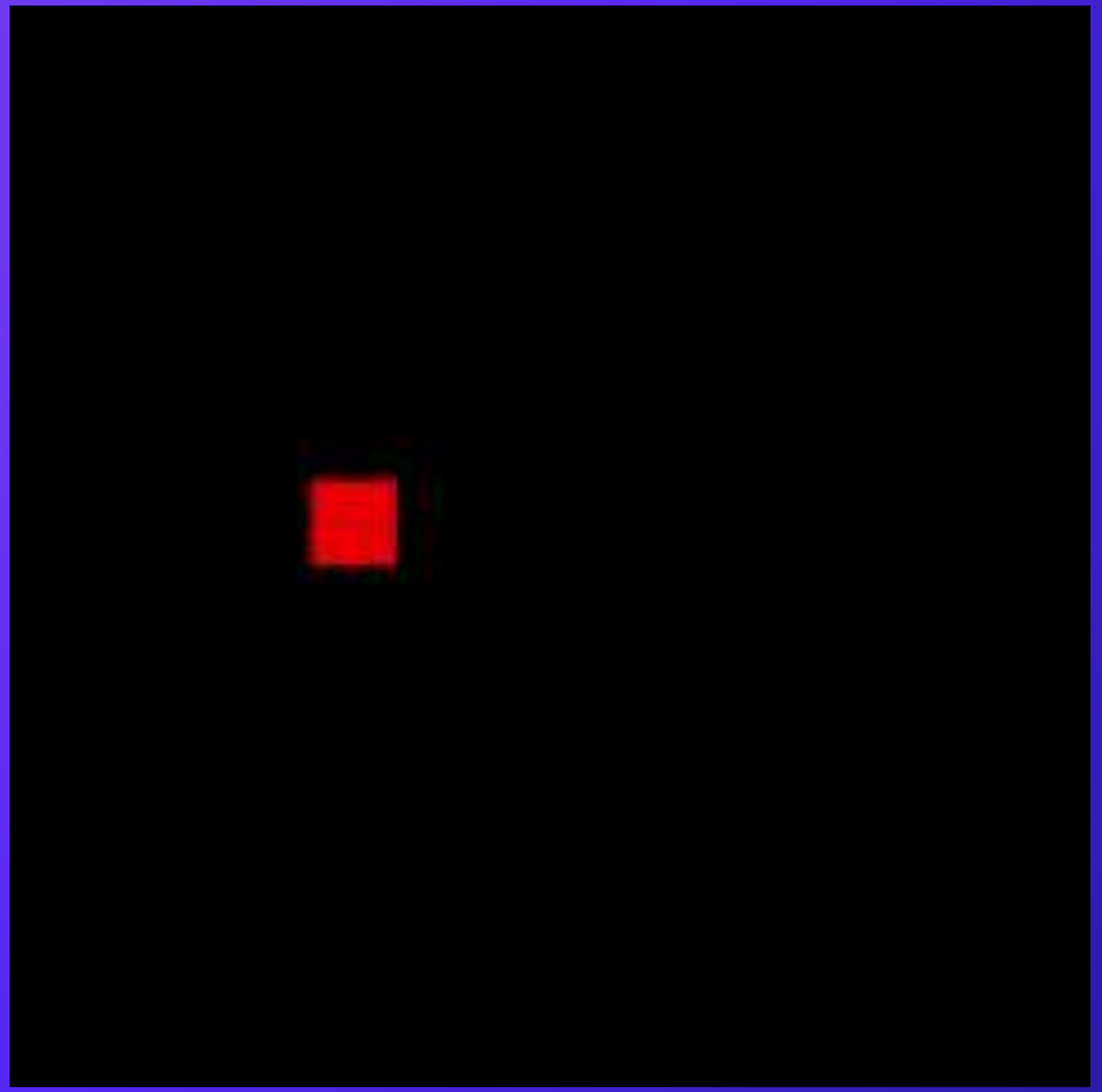
Problem 2: The rectangle should be brought back immediately instead of the user having to move it

Solution: If-else loop to make it go to the other extreme.

```
if x > SCREENX:
 x = 0 #right to left
elif x < 0:
 x = SCREENX #left to right
elif y < 0:
 y = SCREENY #up to down
elif y > SCREENY:
 y = 0 #down to up
```

```python
import pygame
pygame.init()
SCREENX, SCREENY = 500, 500

window = pygame.display.set_mode((SCREENX,SCREENY))
pygame.display.set_caption("first")
x, y, width, height = 250, 250, 50, 50
velocity = 5

jump = False
JUMP_COUNT_MAX = 5
jumpcount = JUMP_COUNT_MAX

run = True
while run:
 pygame.time.delay(20)
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
     run = False
     keys = pygame.key.get_pressed()
 if keys[pygame.K_LEFT]:
     x -= velocity
 elif keys[pygame.K_RIGHT]:
     x += velocity
 elif keys[pygame.K_UP]:
     y -= velocity
 elif keys[pygame.K_DOWN]:
     y += velocity
 elif keys[pygame.K_SPACE]:
     jump = True
```

```python
if x > SCREENX:
    x = 0
elif x < 0:
    x = SCREENX
elif y < 0:
    y = SCREENY
elif y > SCREENY:
    y = 0

if jump:
    if jumpcount >= -JUMP_COUNT_MAX:
        if jumpcount < 0:
            sign = -1
        else:
            sign = 1
        y -= sign * jumpcount**2
        jumpcount -= 1
else:
    jump = False
    jumpcount = JUMP_COUNT_MAX

window.fill("black")
pygame.draw.rect(window,(255,0,255),(x,y,width,height))
pygame.display.update()
pygame.quit()
```



DAY THREE:

Pygame third session

Jump Function

```
jump = False
jumpcount = 10

#insert usual code (keys movement)
    jump = True    #inside the while loop

if jump:
        if jumpcount >= -10:
            y -= (jumpcount**2) * 0.5
            jumpcount -= 1
    else:
        jump = False
        jumpcount = 10
```

Problem: It jumps up twice, but it should jump up and then come back

Reason: It's getting squared, but we need to make it negative to make the block go down.

Solution: add if-else to make it negative if jump count is less than 1 and positive if it is greater than one.

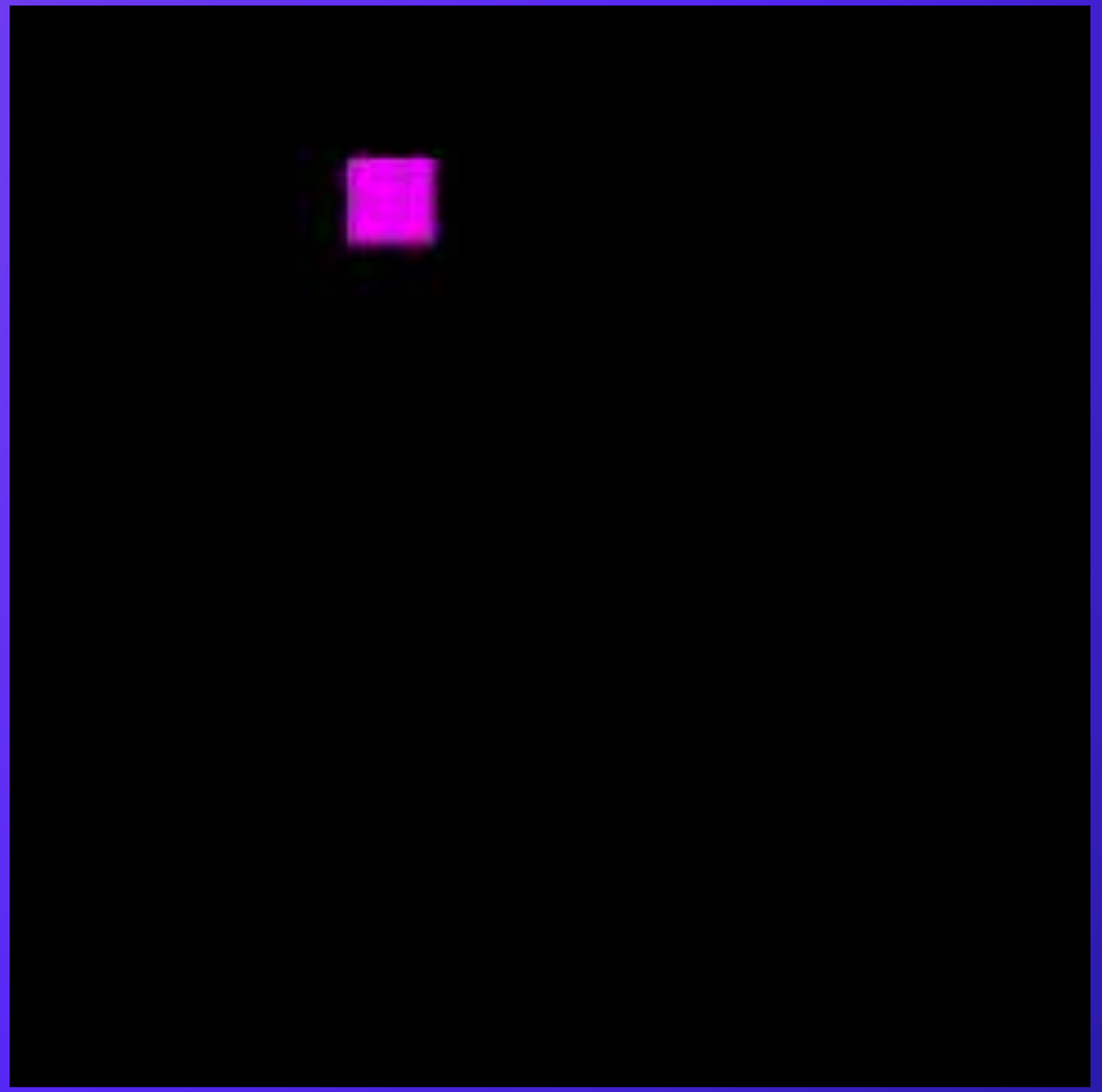
```
if jump:
    if jumpcount >= -10:
        if jumpcount < 0:
            sign = -1
        else:
            sign = 1
        y -= (jumpcount**2) * 0.5 * sign
        jumpcount -= 1
else:
    jump = False
    jumpcount = 10
```

Concept

- Parabolic equation:

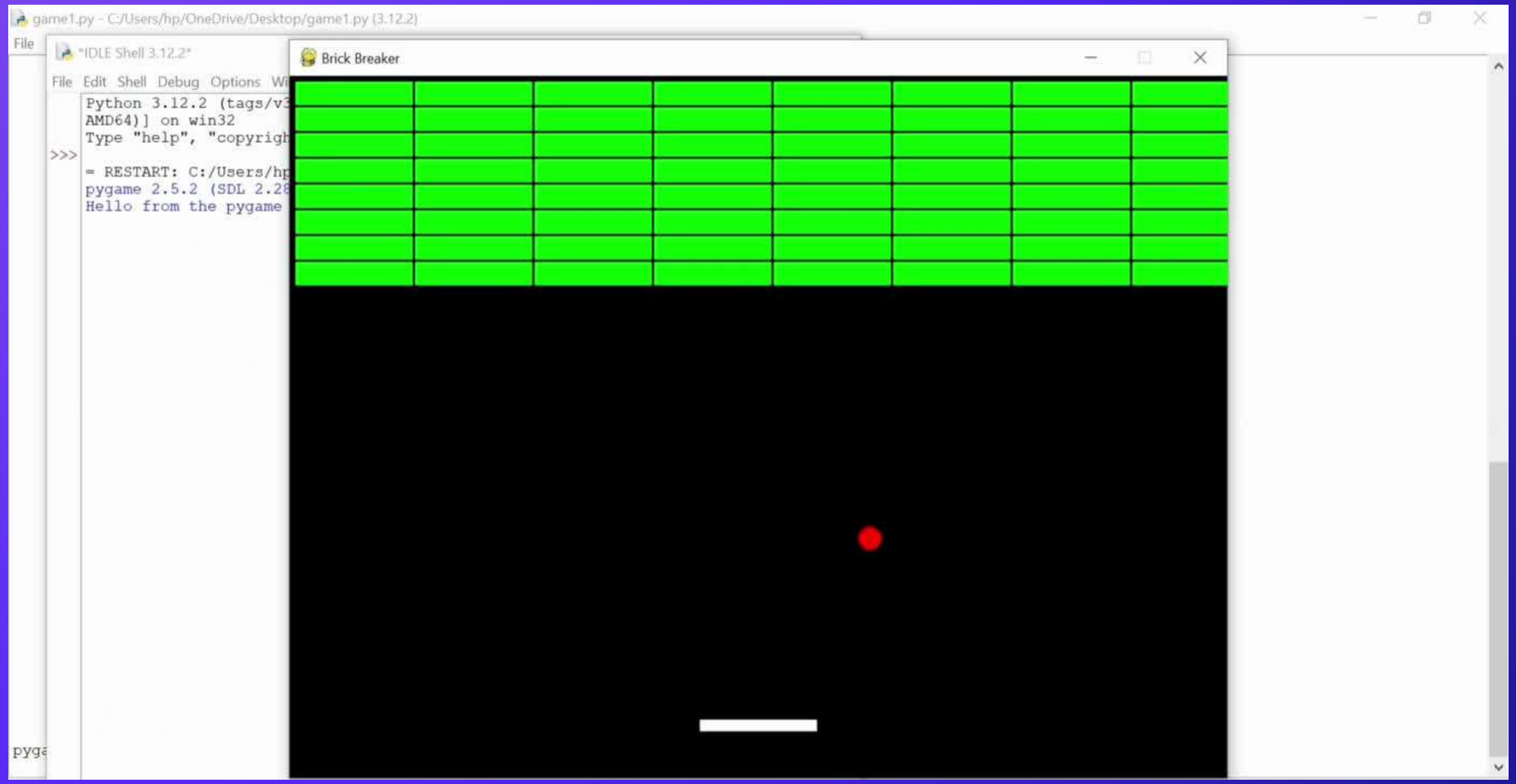
$$y = x^2$$

- Starts from a, goes till -a
- If jumpcount is between a and -a, the jump is in progress
- If jumpcount is -a, it has stopped.



BRICK BREAKER GAME

BRICK BREAKER GAME - A DEMO



CODE OF THE GAME

```
import pygame

# Initialize Pygame
pygame.init()

# Set up the screen
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Brick Breaker")

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
YELLOW = (255, 255, 0)

# Paddle properties
paddle_width = 100
paddle_height = 10
paddle_speed = 20
```

CODE OF THE GAME

```
# Ball properties
ball_radius = 10
ball_speed_x = 5
ball_speed_y = 5

# Brick properties
brick_width = 100
brick_height = 20
bricks = []

# Create paddle
paddle = pygame.Rect(screen_width // 2 - paddle_width // 2, screen_height - 50, paddle_width,
paddle_height)

# Create ball
ball = pygame.Rect(screen_width // 2 - ball_radius, screen_height // 2 - ball_radius, ball_radius *
2, ball_radius * 2)

# Create bricks
for i in range(8):
    for j in range(10):
        brick = pygame.Rect(j * (brick_width + 2) + 5, i * (brick_height + 2) + 5, brick_width,
brick_height)
        bricks.append(brick)

# Ball movement
ball_dx = ball_speed_x
ball_dy = ball_speed_y

# Main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

CODE OF THE GAME

```
    # Move paddle
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and paddle.left > 0:
        paddle.x -= paddle_speed
    if keys[pygame.K_RIGHT] and paddle.right < screen_width:
        paddle.x += paddle_speed

    # Move ball
    ball.x += ball_dx
    ball.y += ball_dy

    # Ball collision with walls
    if ball.top <= 0:
        ball_dy *= -1
    if ball.left <= 0 or ball.right >= screen_width:
        ball_dx *= -1

    # Ball collision with paddle
    if ball.colliderect(paddle) and ball_dy > 0:
        ball_dy *= -1

    # Ball collision with bricks
    for brick in bricks:
        if ball.colliderect(brick):
            bricks.remove(brick)
            ball_dy *= -1

    # Draw everything
    screen.fill(BLACK)
    pygame.draw.rect(screen, WHITE, paddle)
    pygame.draw.ellipse(screen, RED, ball)
    for brick in bricks:
        pygame.draw.rect(screen, GREEN, brick)
    pygame.display.flip()

    # Check if all bricks are destroyed
    if len(bricks) == 0:
        running = False

    # Cap the frame rate
    pygame.time.Clock().tick(60)

pygame.quit()
```