```
import pygame
import requests

pygame.init()

width = 500
height = 635
screen = pygame.display.set_mode((width, height))

pygame.display.set_caption("Sudoku by Arsh Saxena")
icon = pygame.image.load('resources/icon.png')
pygame.display.set_icon(icon)

background = pygame.image.load('resources/menu.png')
font_bold_18 = pygame.font.Font('resources/sf-bold.otf', 18)
font_18 = pygame.font.Font('resources/sf-reg.otf', 18)
font_30 = pygame.font.Font('resources/sf-reg.otf', 30)
font_bold_30 = pygame.font.Font('resources/sf-bold.otf', 30)
font_40 = pygame.font.Font('resources/sf-mono.otf', 40)

level = 1
x = 0
y = 0
dif = 500 / 9
value = 0

def easy():
    global x, y, dif, value

    response = requests.get("https://sugoku.herokuapp.com/board?difficulty=easy")
    grid = response.json()['board']
    """
    If you don't want random values just remove or comment lines 30, 31, and 238 and uncomment lines 35-45 and 239-249.
    """
    # grid = [
    #     [7, 8, 0, 4, 0, 0, 1, 2, 0],
    #     [6, 0, 0, 0, 7, 5, 0, 0, 9],
    #     [0, 0, 0, 6, 0, 1, 0, 7, 8],
    #     [0, 0, 7, 0, 4, 0, 2, 6, 0],
    #     [0, 0, 1, 0, 5, 0, 9, 3, 0],
    #     [9, 0, 4, 0, 6, 0, 0, 0, 5],
    #     [0, 7, 0, 3, 0, 0, 0, 1, 2],
    #     [1, 2, 0, 0, 0, 7, 4, 0, 0],
    #     [0, 4, 9, 2, 0, 6, 0, 0, 7]
    # ]

    def get_cord(pos):
```

```
    global x, y
    x = pos[0] // dif
    y = pos[1] // dif


def draw_box():
    for i in range(2):
        pygame.draw.line(screen, (255, 153, 0), (x * dif - 3, (y + i) * dif), (x * dif + dif + 3, (y + i) * dif), 7)
        pygame.draw.line(screen, (255, 153, 0), ((x + i) * dif, y * dif), ((x + i) * dif, y * dif + dif), 7)


def draw():
    for i in range(9):
        for j in range(9):
            if grid[i][j] != 0:
                pygame.draw.rect(screen, (0, 200, 0), (i * dif, j * dif, dif + 1, dif + 1))
                text1 = font_40.render(str(grid[i][j]), 1, (255, 255, 255))
                screen.blit(text1, (i * dif + 15, j * dif + 4))

    for i in range(10):
        if i % 3 == 0:
            thick = 7
        else:
            thick = 1
        pygame.draw.line(screen, (0, 0, 0), (0, i * dif), (500, i * dif), thick)
        pygame.draw.line(screen, (0, 0, 0), (i * dif, 0), (i * dif, 500), thick)


def draw_value(value):
    text1 = font_40.render(str(value), 1, (0, 0, 0))
    screen.blit(text1, (x * dif + 15, y * dif + 15))


def raise_error1():
    text1 = font_40.render("Wrong!", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def raise_error2():
    text1 = font_40.render("Wrong, not a valid key.", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def valid(m, i, j, value):
    for it in range(9):
        if m[i][it] == value:
            return False
        if m[it][j] == value:
```

```
            return False

    it = i // 3
    jt = j // 3

    for i in range(it * 3, it * 3 + 3):
        for j in range(jt * 3, jt * 3 + 3):
            if m[i][j] == value:
                return False
    return True


def solve(grid, i, j):
    while grid[i][j] != 0:
        if i < 8:
            i += 1
        elif i == 8 and j < 8:
            i = 0
            j += 1
        elif i == 8 and j == 8:
            return True

    pygame.event.pump()

    for it in range(1, 10):
        if valid(grid, i, j, it) == True:
            grid[i][j] = it
            global x, y
            x = i
            y = j

            screen.fill((255, 255, 255))
            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(20)

            if solve(grid, i, j) == 1:
                return True
            else:
                grid[i][j] = 0
            screen.fill((255, 255, 255))

            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(50)
```

```
    return False


def instruction():
    dif_msg = font_18.render("Difficulty: EASY", 1, (0, 0, 0))
    text1 = font_18.render("Press D to reset values to default. Press R to clear", 1, (0, 0, 0))
    text2 = font_18.render("entered values. Press ENTER to visualize.", 1, (0, 0, 0))
    screen.blit(dif_msg, (20, 520))
    screen.blit(text1, (20, 540))
    screen.blit(text2, (20, 560))


def result():
    text1 = font_30.render("FINISHED! QUIT or press R or D.", 1, (0, 0, 0))
    screen.blit(text1, (20, 590))


running = True
flag1 = 0
flag2 = 0
rs = 0
error = 0

while running:
    screen.fill((255, 255, 255))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            pygame.quit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            flag1 = 1
            pos = pygame.mouse.get_pos()
            get_cord(pos)

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                x -= 1
                flag1 = 1
            if event.key == pygame.K_RIGHT:
                x += 1
                flag1 = 1
            if event.key == pygame.K_UP:
                y -= 1
                flag1 = 1
            if event.key == pygame.K_DOWN:
                y += 1
```

```
    flag1 = 1

if event.key == pygame.K_ESCAPE:
    menu()

if event.key == pygame.K_1:
    value = 1
if event.key == pygame.K_2:
    value = 2
if event.key == pygame.K_3:
    value = 3
if event.key == pygame.K_4:
    value = 4
if event.key == pygame.K_5:
    value = 5
if event.key == pygame.K_6:
    value = 6
if event.key == pygame.K_7:
    value = 7
if event.key == pygame.K_8:
    value = 8
if event.key == pygame.K_9:
    value = 9
if event.key == pygame.K_RETURN:
    flag2 = 1

if event.key == pygame.K_r:
    rs = 0
    error = 0
    flag2 = 0
    grid = [
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0]
    ]

if event.key == pygame.K_d:
    rs = 0
    error = 0
    flag2 = 0
    grid = response.json()['board']
    # grid = [
```

```
                    #     [7, 8, 0, 4, 0, 0, 1, 2, 0],
                    #     [6, 0, 0, 0, 7, 5, 0, 0, 9],
                    #     [0, 0, 0, 6, 0, 1, 0, 7, 8],
                    #     [0, 0, 7, 0, 4, 0, 2, 6, 0],
                    #     [0, 0, 1, 0, 5, 0, 9, 3, 0],
                    #     [9, 0, 4, 0, 6, 0, 0, 0, 5],
                    #     [0, 7, 0, 3, 0, 0, 0, 1, 2],
                    #     [1, 2, 0, 0, 0, 7, 4, 0, 0],
                    #     [0, 4, 9, 2, 0, 6, 0, 0, 7]
                    # ]

        if flag2 == 1:
            if solve(grid, 0, 0) == False:
                error = 1
            else:
                rs = 1
            flag2 = 0

        if value != 0:
            draw_value(value)

            if valid(grid, int(x), int(y), value) == True:
                grid[int(x)][int(y)] = value
                flag1 = 0
            else:
                grid[int(x)][int(y)] = 0
                raise_error2()
            value = 0

        if error == 1:
            raise_error1()

        if rs == 1:
            result()

        draw()

        if flag1 == 1:
            draw_box()
        instruction()

        pygame.display.update()

    pygame.quit()

def medium():
    global x, y, dif, value
```

```
response = requests.get("https://sugoku.herokuapp.com/board?difficulty=medium")
grid = response.json()['board']
"""
If you don't want random values just remove or comment lines 288, 289, and 497 and uncomment lines 293-303 and 498-508.
"""
# grid = [
#     [7, 8, 0, 4, 0, 0, 1, 2, 0],
#     [6, 0, 0, 0, 7, 5, 0, 0, 9],
#     [0, 0, 0, 6, 0, 1, 0, 7, 8],
#     [0, 0, 7, 0, 4, 0, 2, 6, 0],
#     [0, 0, 1, 0, 5, 0, 9, 3, 0],
#     [9, 0, 4, 0, 6, 0, 0, 0, 5],
#     [0, 7, 0, 3, 0, 0, 0, 1, 2],
#     [1, 2, 0, 0, 0, 7, 4, 0, 0],
#     [0, 4, 9, 2, 0, 6, 0, 0, 7]
# ]

def get_cord(pos):
    global x
    x = pos[0] // dif
    global y
    y = pos[1] // dif


def draw_box():
    for i in range(2):
        pygame.draw.line(screen, (255, 153, 0), (x * dif - 3, (y + i) * dif), (x * dif + dif + 3, (y + i) * dif), 7)
        pygame.draw.line(screen, (255, 153, 0), ((x + i) * dif, y * dif), ((x + i) * dif, y * dif + dif), 7)


def draw():
    for i in range(9):
        for j in range(9):
            if grid[i][j] != 0:
                pygame.draw.rect(screen, (0, 200, 0), (i * dif, j * dif, dif + 1, dif + 1))
                text1 = font_40.render(str(grid[i][j]), 1, (255, 255, 255))
                screen.blit(text1, (i * dif + 15, j * dif + 4))

    for i in range(10):
        if i % 3 == 0:
            thick = 7
        else:
            thick = 1
        pygame.draw.line(screen, (0, 0, 0), (0, i * dif), (500, i * dif), thick)
        pygame.draw.line(screen, (0, 0, 0), (i * dif, 0), (i * dif, 500), thick)


def draw_value(value):
```

```
    text1 = font_40.render(str(value), 1, (0, 0, 0))
    screen.blit(text1, (x * dif + 15, y * dif + 15))


def raise_error1():
    text1 = font_40.render("Wrong!", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def raise_error2():
    text1 = font_40.render("Wrong, not a valid key.", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def valid(m, i, j, value):
    for it in range(9):
        if m[i][it] == value:
            return False
        if m[it][j] == value:
            return False

    it = i // 3
    jt = j // 3

    for i in range(it * 3, it * 3 + 3):
        for j in range(jt * 3, jt * 3 + 3):
            if m[i][j] == value:
                return False
    return True


def solve(grid, i, j):
    while grid[i][j] != 0:
        if i < 8:
            i += 1
        elif i == 8 and j < 8:
            i = 0
            j += 1
        elif i == 8 and j == 8:
            return True

    pygame.event.pump()

    for it in range(1, 10):
        if valid(grid, i, j, it) == True:
            grid[i][j] = it
            global x, y
            x = i
```

```
            y = j

            screen.fill((255, 255, 255))
            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(20)

            if solve(grid, i, j) == 1:
                return True
            else:
                grid[i][j] = 0
            screen.fill((255, 255, 255))

            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(50)

    return False


def instruction():
    dif_msg = font_18.render("Difficulty: MEDIUM", 1, (0, 0, 0))
    text1 = font_18.render("Press D to reset values to default. Press R to clear", 1, (0, 0, 0))
    text2 = font_18.render("entered values. Press ENTER to visualize.", 1, (0, 0, 0))
    screen.blit(dif_msg, (20, 520))
    screen.blit(text1, (20, 540))
    screen.blit(text2, (20, 560))


def result():
    text1 = font_30.render("FINISHED! QUIT or press R or D.", 1, (0, 0, 0))
    screen.blit(text1, (20, 590))


running = True
flag1 = 0
flag2 = 0
rs = 0
error = 0

while running:
    screen.fill((255, 255, 255))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```
    pygame.quit()

if event.type == pygame.MOUSEBUTTONDOWN:
    flag1 = 1
    pos = pygame.mouse.get_pos()
    get_cord(pos)

if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
        x -= 1
        flag1 = 1
    if event.key == pygame.K_RIGHT:
        x += 1
        flag1 = 1
    if event.key == pygame.K_UP:
        y -= 1
        flag1 = 1
    if event.key == pygame.K_DOWN:
        y += 1
        flag1 = 1

    if event.key == pygame.K_ESCAPE:
        menu()

    if event.key == pygame.K_1:
        value = 1
    if event.key == pygame.K_2:
        value = 2
    if event.key == pygame.K_3:
        value = 3
    if event.key == pygame.K_4:
        value = 4
    if event.key == pygame.K_5:
        value = 5
    if event.key == pygame.K_6:
        value = 6
    if event.key == pygame.K_7:
        value = 7
    if event.key == pygame.K_8:
        value = 8
    if event.key == pygame.K_9:
        value = 9
    if event.key == pygame.K_RETURN:
        flag2 = 1

    if event.key == pygame.K_r:
        rs = 0
        error = 0
```

```
            flag2 = 0
            grid = [
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0]
            ]

        if event.key == pygame.K_d:
            rs = 0
            error = 0
            flag2 = 0
            grid = response.json()['board']
            # grid = [
            #     [7, 8, 0, 4, 0, 0, 1, 2, 0],
            #     [6, 0, 0, 0, 7, 5, 0, 0, 9],
            #     [0, 0, 0, 6, 0, 1, 0, 7, 8],
            #     [0, 0, 7, 0, 4, 0, 2, 6, 0],
            #     [0, 0, 1, 0, 5, 0, 9, 3, 0],
            #     [9, 0, 4, 0, 6, 0, 0, 0, 5],
            #     [0, 7, 0, 3, 0, 0, 0, 1, 2],
            #     [1, 2, 0, 0, 0, 7, 4, 0, 0],
            #     [0, 4, 9, 2, 0, 6, 0, 0, 7]
            # ]

if flag2 == 1:
    if solve(grid, 0, 0) == False:
        error = 1
    else:
        rs = 1
    flag2 = 0

if value != 0:
    draw_value(value)

    if valid(grid, int(x), int(y), value) == True:
        grid[int(x)][int(y)] = value
        flag1 = 0
    else:
        grid[int(x)][int(y)] = 0
        raise_error2()
    value = 0
```

```
        if error == 1:
            raise_error1()

        if rs == 1:
            result()

        draw()

        if flag1 == 1:
            draw_box()
        instruction()

        pygame.display.update()

    pygame.quit()

def hard():
    global x, y, dif, value

    response = requests.get("https://sugoku.herokuapp.com/board?difficulty=hard")
    grid = response.json()['board']
    """
    If you don't want random values just remove or comment lines 547, 548, and 756 and uncomment lines 552-562 and 757-767.
    """
    # grid = [
    #     [7, 8, 0, 4, 0, 0, 1, 2, 0],
    #     [6, 0, 0, 0, 7, 5, 0, 0, 9],
    #     [0, 0, 0, 6, 0, 1, 0, 7, 8],
    #     [0, 0, 7, 0, 4, 0, 2, 6, 0],
    #     [0, 0, 1, 0, 5, 0, 9, 3, 0],
    #     [9, 0, 4, 0, 6, 0, 0, 0, 5],
    #     [0, 7, 0, 3, 0, 0, 0, 1, 2],
    #     [1, 2, 0, 0, 0, 7, 4, 0, 0],
    #     [0, 4, 9, 2, 0, 6, 0, 0, 7]
    # ]

    def get_cord(pos):
        global x
        x = pos[0] // dif
        global y
        y = pos[1] // dif


    def draw_box():
        for i in range(2):
            pygame.draw.line(screen, (255, 153, 0), (x * dif - 3, (y + i) * dif), (x * dif + dif + 3, (y + i) * dif), 7)
            pygame.draw.line(screen, (255, 153, 0), ((x + i) * dif, y * dif), ((x + i) * dif, y * dif + dif), 7)
```

```
def draw():
    for i in range(9):
        for j in range(9):
            if grid[i][j] != 0:
                pygame.draw.rect(screen, (0, 200, 0), (i * dif, j * dif, dif + 1, dif + 1))
                text1 = font_40.render(str(grid[i][j]), 1, (255, 255, 255))
                screen.blit(text1, (i * dif + 15, j * dif + 4))

    for i in range(10):
        if i % 3 == 0:
            thick = 7
        else:
            thick = 1
        pygame.draw.line(screen, (0, 0, 0), (0, i * dif), (500, i * dif), thick)
        pygame.draw.line(screen, (0, 0, 0), (i * dif, 0), (i * dif, 500), thick)


def draw_value(value):
    text1 = font_40.render(str(value), 1, (0, 0, 0))
    screen.blit(text1, (x * dif + 15, y * dif + 15))


def raise_error1():
    text1 = font_40.render("Wrong!", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def raise_error2():
    text1 = font_40.render("Wrong, not a valid key.", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))


def valid(m, i, j, value):
    for it in range(9):
        if m[i][it] == value:
            return False
        if m[it][j] == value:
            return False

    it = i // 3
    jt = j // 3

    for i in range(it * 3, it * 3 + 3):
        for j in range(jt * 3, jt * 3 + 3):
            if m[i][j] == value:
                return False
    return True
```

```
def solve(grid, i, j):
    while grid[i][j] != 0:
        if i < 8:
            i += 1
        elif i == 8 and j < 8:
            i = 0
            j += 1
        elif i == 8 and j == 8:
            return True

    pygame.event.pump()

    for it in range(1, 10):
        if valid(grid, i, j, it) == True:
            grid[i][j] = it
            global x, y
            x = i
            y = j

            screen.fill((255, 255, 255))
            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(20)

            if solve(grid, i, j) == 1:
                return True
            else:
                grid[i][j] = 0
            screen.fill((255, 255, 255))

            draw()
            draw_box()
            pygame.display.update()
            pygame.time.delay(50)

    return False


def instruction():
    dif_msg = font_18.render("Difficulty: HARD", 1, (0, 0, 0))
    text1 = font_18.render("Press D to reset values to default. Press R to clear", 1, (0, 0, 0))
    text2 = font_18.render("entered values. Press ENTER to visualize.", 1, (0, 0, 0))
    screen.blit(dif_msg, (20, 520))
    screen.blit(text1, (20, 540))
    screen.blit(text2, (20, 560))
```

```
def result():
    text1 = font_30.render("FINISHED! QUIT or press R or D.", 1, (0, 0, 0))
    screen.blit(text1, (20, 590))


running = True
flag1 = 0
flag2 = 0
rs = 0
error = 0

while running:
    screen.fill((255, 255, 255))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            pygame.quit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            flag1 = 1
            pos = pygame.mouse.get_pos()
            get_cord(pos)

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                x -= 1
                flag1 = 1
            if event.key == pygame.K_RIGHT:
                x += 1
                flag1 = 1
            if event.key == pygame.K_UP:
                y -= 1
                flag1 = 1
            if event.key == pygame.K_DOWN:
                y += 1
                flag1 = 1

            if event.key == pygame.K_ESCAPE:
                menu()

            if event.key == pygame.K_1:
                value = 1
            if event.key == pygame.K_2:
                value = 2
            if event.key == pygame.K_3:
```

```
    value = 3
if event.key == pygame.K_4:
    value = 4
if event.key == pygame.K_5:
    value = 5
if event.key == pygame.K_6:
    value = 6
if event.key == pygame.K_7:
    value = 7
if event.key == pygame.K_8:
    value = 8
if event.key == pygame.K_9:
    value = 9
if event.key == pygame.K_RETURN:
    flag2 = 1

if event.key == pygame.K_r:
    rs = 0
    error = 0
    flag2 = 0
    grid = [
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0]
    ]

if event.key == pygame.K_d:
    rs = 0
    error = 0
    flag2 = 0
    grid = response.json()['board']
    # grid = [
    #     [7, 8, 0, 4, 0, 0, 1, 2, 0],
    #     [6, 0, 0, 0, 7, 5, 0, 0, 9],
    #     [0, 0, 0, 6, 0, 1, 0, 7, 8],
    #     [0, 0, 7, 0, 4, 0, 2, 6, 0],
    #     [0, 0, 1, 0, 5, 0, 9, 3, 0],
    #     [9, 0, 4, 0, 6, 0, 0, 0, 5],
    #     [0, 7, 0, 3, 0, 0, 0, 1, 2],
    #     [1, 2, 0, 0, 0, 7, 4, 0, 0],
    #     [0, 4, 9, 2, 0, 6, 0, 0, 7]
    # ]
```

```
        if flag2 == 1:
            if solve(grid, 0, 0) == False:
                error = 1
            else:
                rs = 1
            flag2 = 0

        if value != 0:
            draw_value(value)

            if valid(grid, int(x), int(y), value) == True:
                grid[int(x)][int(y)] = value
                flag1 = 0
            else:
                grid[int(x)][int(y)] = 0
                raise_error2()
            value = 0

        if error == 1:
            raise_error1()

        if rs == 1:
            result()

        draw()

        if flag1 == 1:
            draw_box()
        instruction()

        pygame.display.update()

    pygame.quit()

def menu():
    global level
    MainRun = True
    while MainRun:
        screen.fill((18, 18, 18))
        screen.blit(background, (0, 0))

        welcome_line = font_bold_30.render("WELCOME", 1, (255, 255, 255))
        line1 = font_bold_18.render("• For EASY difficulty sudoku, please press 1.", 1, (0, 255, 0))
        line2 = font_bold_18.render("• For MEDIUM difficulty sudoku, please press 2.", 1, (242, 255, 0))
        line3 = font_bold_18.render("• For HARD difficulty sudoku, please press 3.", 1, (255, 0, 0))
        screen.blit(welcome_line, (170, 510))
        screen.blit(line1, (20, 550))
```

```
        screen.blit(line2, (20, 575))
        screen.blit(line3, (20, 600))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                MainRun = False

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_1:
                    level = 1
                    MainRun = False
                    print("Difficulty: Easy")
                    easy()

                if event.key == pygame.K_2:
                    level = 2
                    MainRun = False
                    print("Difficulty: Medium")
                    medium()

                if event.key == pygame.K_3:
                    level = 3
                    MainRun = False
                    print("Difficulty: Hard")
                    hard()

        pygame.display.update()

menu()

pygame.display.update()
```