**Task 12. Simulate Gaming concepts using Pygame CO5-K5**

Aim**:**

To Simulate Gaming concepts using Pygame

**SnakeGame:**

**Problem 1.Write a python program to create a snakeGame using pygame package.**

**Conditions:**

1.Set the window size

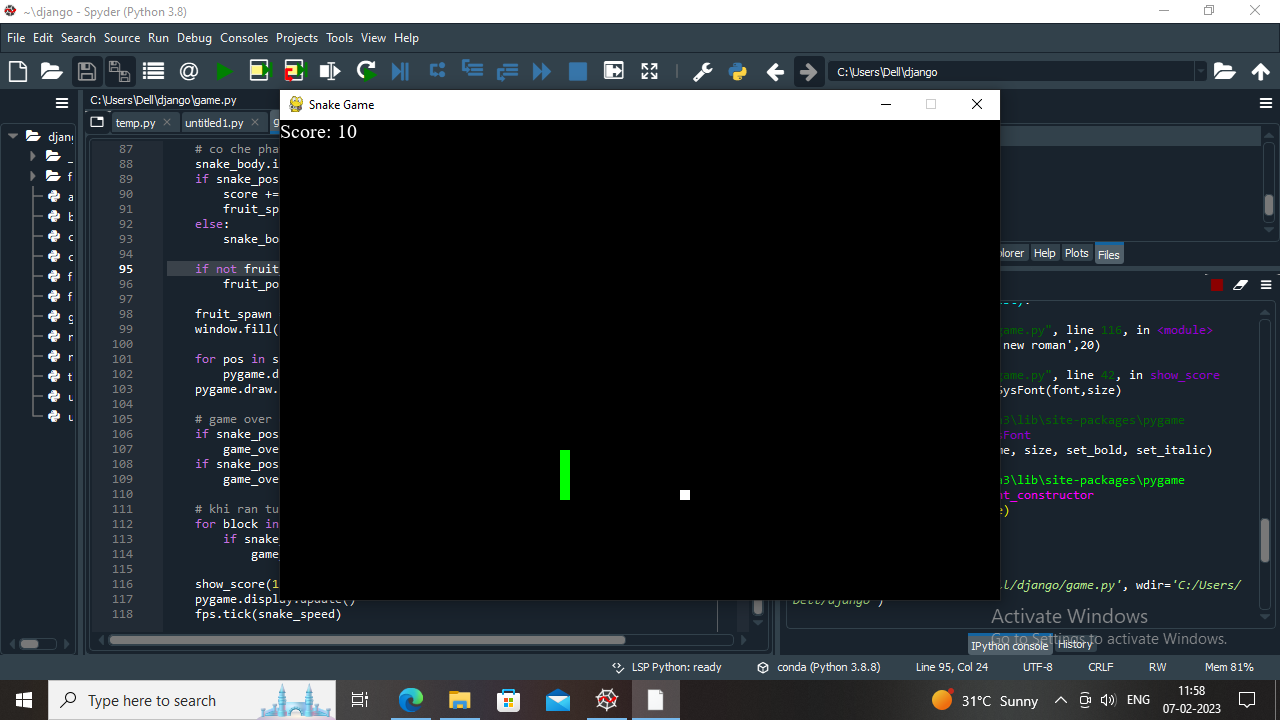
2.Create a snake

3.Make the snake to move in the directions when left,right,down and up key is pressed

4.When the snake hits the fruit.increase the score by 10

5.If the snake hits the window.Game over

Sample Output:



**Algorithm:**

1. Import pygame package and initialize it
2. Define the window size and title
3. Create a Snake class which initializes the snake position, color, and movement
4. Create a Fruit class which initializes the fruit position and color
5. Create a function to check if the snake collides with the fruit and increase the score
6. Create a function to check if the snake collides with the window and end the game
7. Create a function to update the snake position based on the user input
8. Create a function to update the game display and draw the snake and fruit
9. Create a game loop to continuously update the game display, snake position, and check for collisions
10. End the game if the user quits or the snake collides with the window

**Program:**

# importing libraries

import pygame

import time

import random

snake\_speed = 15

# Window size

window\_x = 720

window\_y = 480

# defining colors

black = pygame.Color(0, 0, 0)

white = pygame.Color(255, 255, 255)

red = pygame.Color(255, 0, 0)

green = pygame.Color(0, 255, 0)

blue = pygame.Color(0, 0, 255)

# Initialising pygame

pygame.init()

# Initialise game window

pygame.display.set\_caption('GeeksforGeeks Snakes')

game\_window = pygame.display.set\_mode((window\_x, window\_y))

# FPS (frames per second) controller

fps = pygame.time.Clock()

# defining snake default position

snake\_position = [100, 50]

# defining first 4 blocks of snake body

snake\_body = [[100, 50],

[90, 50],

[80, 50],

[70, 50]

]

# fruit position

fruit\_position = [random.randrange(1, (window\_x//10)) \* 10,

random.randrange(1, (window\_y//10)) \* 10]

fruit\_spawn = True

# setting default snake direction towards

# right

direction = 'RIGHT'

change\_to = direction

# initial score

score = 0

# displaying Score function

def show\_score(choice, color, font, size):

# creating font object score\_font

score\_font = pygame.font.SysFont(font, size)

# create the display surface object

# score\_surface

score\_surface = score\_font.render('Score : ' + str(score), True, color)

# create a rectangular object for the text

# surface object

score\_rect = score\_surface.get\_rect()

# displaying text

game\_window.blit(score\_surface, score\_rect)

# game over function

def game\_over():

# creating font object my\_font

my\_font = pygame.font.SysFont('times new roman', 50)

# creating a text surface on which text

# will be drawn

game\_over\_surface = my\_font.render(

'Your Score is : ' + str(score), True, red)

# create a rectangular object for the text

# surface object

game\_over\_rect = game\_over\_surface.get\_rect()

# setting position of the text

game\_over\_rect.midtop = (window\_x/2, window\_y/4)

# blit will draw the text on screen

game\_window.blit(game\_over\_surface, game\_over\_rect)

pygame.display.flip()

# after 2 seconds we will quit the program

time.sleep(2)

# deactivating pygame library

pygame.quit()

# quit the program

quit()

# Main Function

while True:

# handling key events

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_UP:

change\_to = 'UP'

if event.key == pygame.K\_DOWN:

change\_to = 'DOWN'

if event.key == pygame.K\_LEFT:

change\_to = 'LEFT'

if event.key == pygame.K\_RIGHT:

change\_to = 'RIGHT'

# If two keys pressed simultaneously

# we don't want snake to move into two

# directions simultaneously

if change\_to == 'UP' and direction != 'DOWN':

direction = 'UP'

if change\_to == 'DOWN' and direction != 'UP':

direction = 'DOWN'

if change\_to == 'LEFT' and direction != 'RIGHT':

direction = 'LEFT'

if change\_to == 'RIGHT' and direction != 'LEFT':

direction = 'RIGHT'

# Moving the snake

if direction == 'UP':

snake\_position[1] -= 10

if direction == 'DOWN':

snake\_position[1] += 10

if direction == 'LEFT':

snake\_position[0] -= 10

if direction == 'RIGHT':

snake\_position[0] += 10

# Snake body growing mechanism

# if fruits and snakes collide then scores

# will be incremented by 10

snake\_body.insert(0, list(snake\_position))

if snake\_position[0] == fruit\_position[0] and snake\_position[1] == fruit\_position[1]:

score += 10

fruit\_spawn = False

else:

snake\_body.pop()

if not fruit\_spawn:

fruit\_position = [random.randrange(1, (window\_x//10)) \* 10,

random.randrange(1, (window\_y//10)) \* 10]

fruit\_spawn = True

game\_window.fill(black)

for pos in snake\_body:

pygame.draw.rect(game\_window, green,

pygame.Rect(pos[0], pos[1], 10, 10))

pygame.draw.rect(game\_window, white, pygame.Rect(

fruit\_position[0], fruit\_position[1], 10, 10))

# Game Over conditions

if snake\_position[0] < 0 or snake\_position[0] > window\_x-10:

game\_over()

if snake\_position[1] < 0 or snake\_position[1] > window\_y-10:

game\_over()

# Touching the snake body

for block in snake\_body[1:]:

if snake\_position[0] == block[0] and snake\_position[1] == block[1]:

game\_over()

# displaying score continuously

show\_score(1, white, 'times new roman', 20)

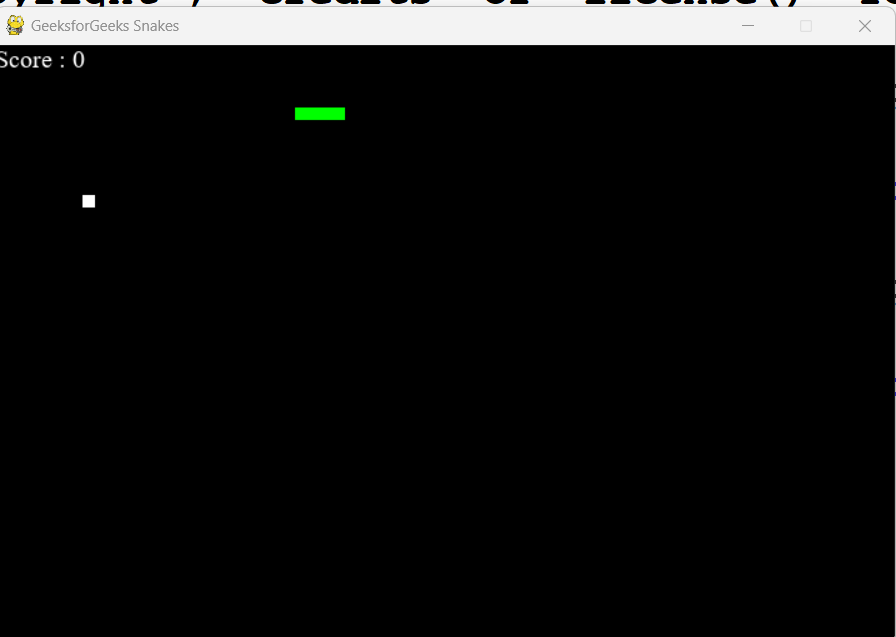
# Refresh game screen

pygame.display.update()

# Frame Per Second /Refresh Rate

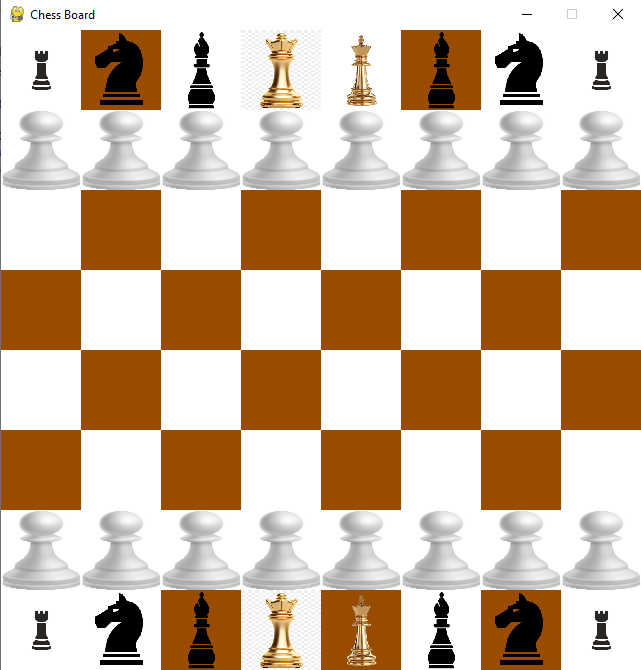
fps.tick(snake\_speed)

Output

****

**Problem 2.Write a python program to Develop a chess board using pygame.**

**Sample output:**



**Algorithm:**

1. Import pygame and initialize it.
2. Set screen size and title.
3. Define colors for the board and pieces.

Define a function to draw the board by looping over rows and columns and drawing squares of different colors.

1. Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding square.
2. Define the initial state of the board as a list of lists containing the pieces.
3. Draw the board and pieces on the screen.
4. Start the game loop.

**Program:**

import pygame

# Initialize pygame

pygame.init()

# Set screen size and title

screen\_size = (640, 640)

screen = pygame.display.set\_mode(screen\_size)

pygame.display.set\_caption('Chess Board')

# Define colors

black = (0, 0, 0)

white = (255, 255, 255)

brown = (153, 76, 0)

# Define function to draw the board

def draw\_board():

for row in range(8):

for col in range(8):

square\_color = white if (row + col) % 2 == 0 else brown

square\_rect = pygame.Rect(col \* 80, row \* 80, 80, 80)

pygame.draw.rect(screen, square\_color, square\_rect)

# Define function to draw the pieces

def draw\_pieces(board):

SQUARE\_SIZE = 80 # or whatever your square size is

piece\_images = {

'p': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/pawn.png'), (SQUARE\_SIZE, SQUARE\_SIZE)),

'r': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/rook.jpg'), (SQUARE\_SIZE, SQUARE\_SIZE)),

'n': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/knight.png'), (SQUARE\_SIZE, SQUARE\_SIZE)),

'b': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/bishop.png'), (SQUARE\_SIZE, SQUARE\_SIZE)),

'q': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/queen.png'), (SQUARE\_SIZE, SQUARE\_SIZE)),

'k': pygame.transform.scale(pygame.image.load('E:/SUMMER 25\_26/PYTHON/lab/New folder/king.png'), (SQUARE\_SIZE, SQUARE\_SIZE))

}

for row in range(8):

for col in range(8):

piece = board[row][col]

if piece != '.':

piece\_image = piece\_images[piece]

piece\_rect = pygame.Rect(col \* 80, row \* 80, 80, 80)

screen.blit(piece\_image, piece\_rect)

# Define initial state of the board

board = [

['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'],

['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],

['.', '.', '.', '.', '.', '.', '.', '.'],

['.', '.', '.', '.', '.', '.', '.', '.'],

['.', '.', '.', '.', '.', '.', '.', '.'],

['.', '.', '.', '.', '.', '.', '.', '.'],

['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],

['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r']

]

# Draw board and pieces

draw\_board()

draw\_pieces(board)

# Start game loop

while True:

for event in pygame.event.get():

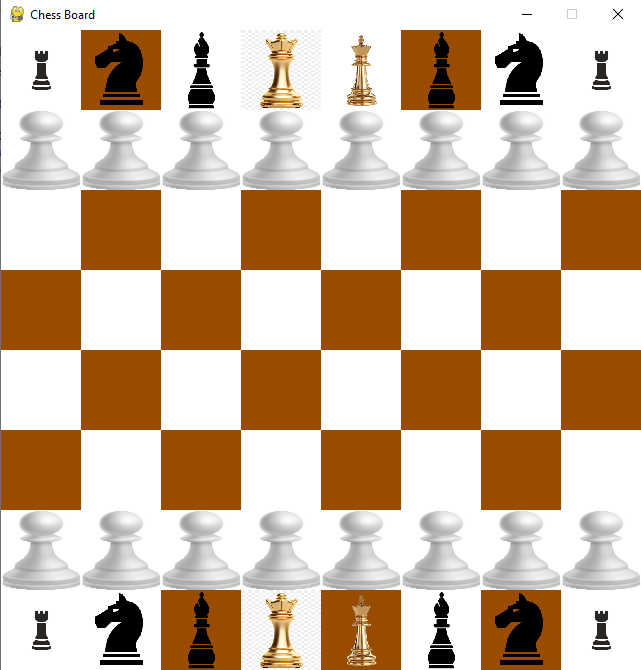
if event.type == pygame.QUIT:

pygame.quit()

quit()

pygame.display.update()

**Output**:



**Result:** Thus the program for pygame is executed and verified successfully.