Dt: 8/10/25 Task 12 - Simulate Gaming Concepts Using Pygame.

Aim:

To write a python program to create a Snake game using Pygame package.

Algorithm:-

1. Set the window size
2. Create 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 increse the score by 10
5. If the Snake hits the window. Game over

```
# importing libraries
import pygame
import time
import random
Snake_speed = 15
# window size
window_x = 720
window_y = 480
# defining colours
black = pygame.Colour(0,0,0)
white = pygame.color(255,255,255)
red = pygame.color(255,0,0)
green = Pygame.colour(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]
```

Output

Score :0

```
#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(
```

```
#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':
```

```
        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[0] < 0 or Snake_position[0] > window_y-10:
        game_over()
```

```
# Touching the snake - body [1:]
for block in snake - body[1:]:
    if snake - position [0] == block [0] and snake_position[1] == block
    [1]:
        game_over()
    # displaying score continuosly
    show_score (1, white, 'times new romans', 20)

# Refresh game screen
Pygame.display.Update()

# frame per second / Refresh Rate
    fps.tick (snake_speed)
```

Task 12.2

Aim:

To write a python Program to develop a chess board using Pygame.

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.
4. Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding squale.
5. Define the initial State of the board as a list of lists containing the peices
6. Draw the board and pieces on the screen
7. Start the game loop.

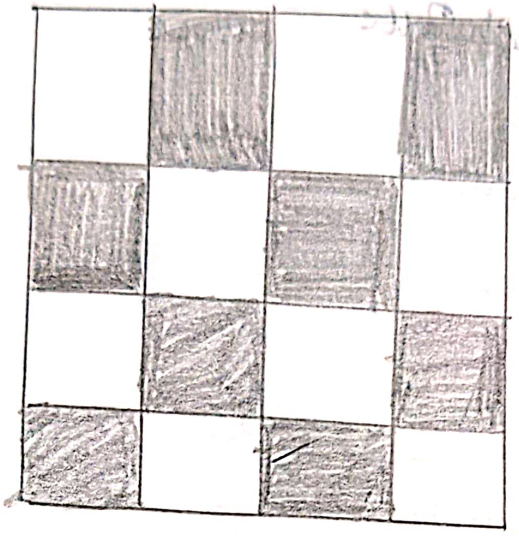
Program:

```
import pygame
#Initialize Pygame
 Pygame.init()
#Set Screen Size and title
  Screen_Size = (640,640)
  Screen  = Pygame.display.set_model (Screen_size)
 Pygame.display.set.Caption ("Chess Board")

 # Define Colours
   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):
```

Output:



```
        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):
    piece_image = {
        'r': pygame.image.load('images/rook.png'),
        'n': pygame.image.load('images/knight.png'),
        'b': pygame.image.load('images/bishop.png'),
        'q': pygame.image.load('images/king.png');
        'p': pygame.image.load('images/pawn.png')
        }
        for row in range(8):
        for col in range(8):
            piece = board[row][col]
            if piece != '.':
                piece_image = piece_image[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 Gameloop
while True:
    for event in Pygame.event.get():
        if event.type == Pygame.QUIT:
            Pygame.quit()
            quit()
        Pygame.display.update()
```

Completed

VEL TECH - CSE	
EX NO.	10
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	2
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	
SIGNATURE DATE	15

Result:

Thus the python program for executing the chess boards was successfully completed.