Task 12 - Simulate Gaming concepts using Pygame.

Aim: To Simulate gaming Concepts using Python.

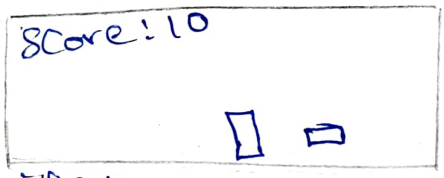
SnakeGame

Problem: write a Python Program to create a snake game.

Conditions:-

1. set the window size
2. Create a snake
3. make the snake to move in the directions such as left, right down and up is pressed.
4. If the snake hits the windows - Game over

Sample output:-



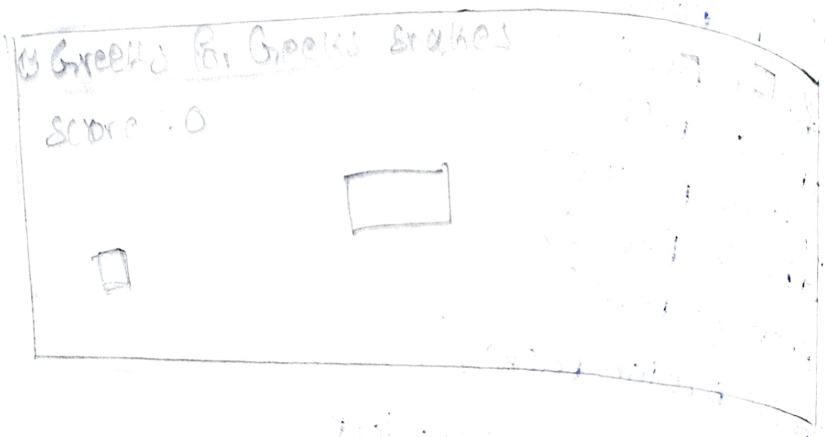
Algorithm:

1. Import Pygame package and initialize it.
2. Define the window size and title.
3. Create a snake class which indicates the snake position, color, and movement.
4. create a function to check if the snake collides with the limit and increase the score.
5. Create a game loop to continously update the game display. Snake position and check for collision
6. 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)
```

Output :-



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

# initialising pygame
pygame.init()
# initialise game window
pygame.display.set_caption('Geeks for Geeks snakes')
game_window = pygame.display.set_mode(window_x, window_y)
# FPS (frame per second) controller
fps = pygame.time.Clock()
# defining snake default position
snake_position = [100, 50]
# defing 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_position = 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 objects score_font
    score_font = pygame.font.SysFont(font, size)
# creating font objects font.
    score_font = pygame.font.SysFont(font, size)
# create a rectangular object for the text
# surface object
    score_surface = score_font.set_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 = 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 the 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_LEFT:
                if event.key == pygame.K_RIGHT:
                if event.key == pygame.K_
# 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'
    if change_to ==
```

```
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 score
# 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()
# Touching the snake body
for block in snake_body[1:]:
        if snake_position[0] == block[0] and snake_position
                                        [1] == block[1]
# displaying score continuously.
    show_score(1, white, 'times new woman', 20)
# Refresh game screen.
    pygame.display.update()
# Frame per second / Refresh Rate
    fps.tick(snake_speed)
```

Problem2:- write a Python program to develop a chess board using Pygame.

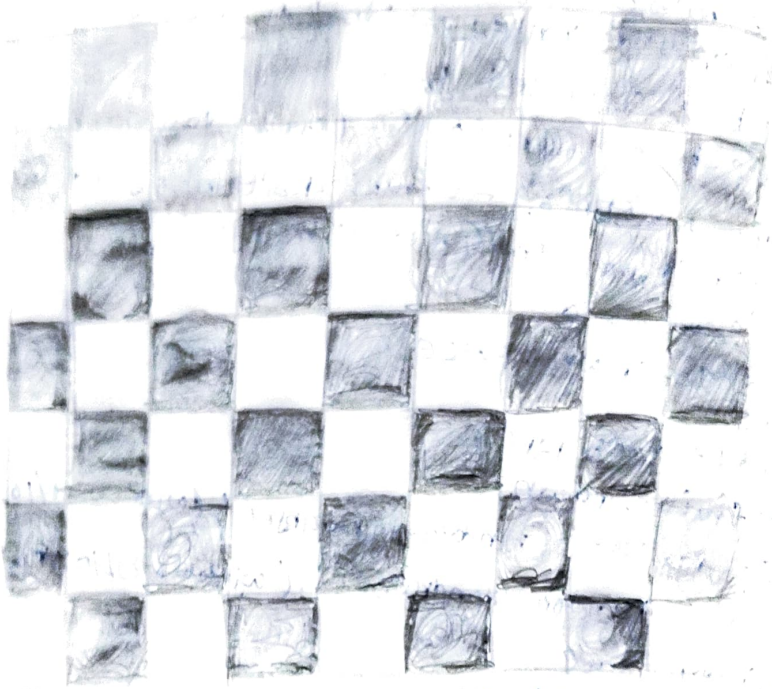
Algorithm:

1. Import Pygame and initialize it.
2. set screen size and title.
3. Define colors for the board and pieces.
4. Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding square.
5. Define the initial state of the board as a list of list containing the pieces.
6. 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 drawn 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, color, square_rect)

# Define function to draw the pieces
def draw_pieces(board):
  piece_images = {
      'r': Pygame.image.load('image/rooks.png'),
      'n': Pygame.image.load('image/knight.png'),
      'b': Pygame.image.load('image/bishop.png'),
```



```
'q': pygame.image.load('images/queen.png'),
'k': pygame.image.load('images/king.png'),
'p': pygame.image.load('image/pawn.png')
}
for row in range(8):
    for col in range(8):
        if piece != '--':
            piece_image = piece_images[piece]
            piece_rect = pygame.Rect(col*80, row*8
            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
            pygame.quit()
            quit()
pygame.display.update()
```

Result: Thus, the program is executed verified successfully

VEL TECH	
EX No.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	8/5/25