TASK 12 : use the Tkinter module for UI design (15/10/25)

AIM : To design & implement a simple Maze Game using Python's Pygame library, where player navigates a square through a maze to reach

Algorithm :

Step 1: Start

Step 2: import required modules:
- import pygame for game development functions.
- import sys

Step 3: inditialize Pygame using Pygame.init()

Step 4 : Setup the game window:
- Define screen width & height
- create the display surface using Pygame.display.set_mode().

Step 5. Define colours using RGB values for white, black, blue, green, and red.

Step 6: Create Player
- Start at (50,50)
- Goal placed at (550,350)

Step 7: Create maze walls:
- Define a list of rectangular wall obstacles

Step 8 : Define a function check_collision (rect, walls):
- check if Player's rectangle collides
- Return True if a collision occurs.

Program :

```
    mport pygame
    importsys
    Pygame. int()
   WIDTH, HEIGHT = 600, 400
   Screen = Pygame, display.set_mode((WIDTH, HEIGHT))
   Pygame. display.set_Caption ("Maze Game")

   WHITE = (255, 255, 255)
   BLACK = (0,0,0)
   BLUE = (0, 0, 200)
   GREEN = (0,200, 0)
   RED = (200,0,0)
   Clock = Pygame . time. clock()
   Player_ size = 20
   Player = Pygame. Rect (50,50, Playen _size, Player_size)

# Goal setup
   goal = Pygame. Rect (550, 350, Playen _ size, Player_size)
  walls = [
     Py.game. Recl (100,0,20,300),
     Py. game. Rect (200, 100,20, 300),
     Py . game Rect (300,0,20,250),
    Py. game Rect (400, 150,20, 250),
    Py . game Rect (500, 0,20,250),
   ]
```

```
def check_collision (rect, walls):
    for wall in walls:
     if rect.colliderect(wall):
        return True
    return False

running = True
while running
    screen. fill (WHITE)

 for event in Pygame. event.get():
        if event.type == Pygame. QUIT:
          running = False
   keys = Pygame. key.get_Pressed()
    move_x, move_y =0, 0
 if keys [Pygame. K_LEFT]:
    move _x = -3
 if keys [Pygame. k_left]:
    move _ x = -3
 if keys [Pygame. K_ RIGHT].
    move _ x = 3
 if keys [Pygame. K_up]:
    move_ y = -3
  if Keys [Pygame. K_ DOWN]:
     move_ y = 3
   Pygame. display. flip()
   Clock. tick (30)
   Pygame. Quit().
    sys. exit()
```

Step 10. game objects

- Draw Walls (black), goal (green) and player (blue).

Step 11: update display using Pygame. display. flip () and maintain frame rate with clock. tick (30).

Step 12. Exit game using Pygame. quit () and sys. exit ()

Step 13. End the Program

Sample output / game Description:

- The window displays a maze made of black walls
- The blue square represents the player.
- The green square represents the goal.
- The Player moves using Arrow keys.
    - ↑ up
    - ↓ Down
    - ← Left
    - → Right
- if Player touches a wall, the moves is undone
- when, the Player reaches the goal, message 'you win"! appears for 2 seconds, and game exits.

VELTECH

EX No.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

[illegible]

RESULT: Hence, the Tkinter, module for UI design has been Executed Successfully