Task 2:

Identify a problem statement within your choosen domain and device a solution for it. Then articulate the problem and your proposed solution

Answer:

Problem Statement:

In the Python Snake game, there is no clear win condition or game-over state that signifies the player's victory or defeat, making it difficult for players to determine when they have won or lost the game.

Proposed Solution:

Problem Articulation:

Currently, the game continues indefinitely without an endpoint, and players may not know when they've achieved a high score or when the game is over. This can lead to confusion and a lack of a sense of accomplishment.

Proposed Solution Articulation:

To make the game more engaging and provide a clear win or loss condition, we should add a win and lose state. Here's how to implement it:

1. *Problem:* Lack of a game-over state

   Solution: Implement a game-over condition when the Snake collides with the screen border or itself. When this happens, display a "Game Over" message and allow the player to restart the game.

2. *Problem:* Lack of a win condition

   Solution: Set a target score that, when achieved, signifies a win. When the player reaches this score, display a "You Win" message and allow the player to restart the game.

3. *Problem:* No feedback for a high score

Solution: Keep track of the player's highest score during their gameplay. Whenever they set a new high score, display a congratulatory message to acknowledge their achievement.

By adding these elements, players will have a clear understanding of when they've succeeded or failed in the game, creating a more satisfying and enjoyable gaming experience.

Solution:

```
# Game over text
def Game_Over():

    game_over = True
    while game_over:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    game_over = False
                elif event.key == pygame.K_ESCAPE:
                    pygame.quit()
                    quit()

        pygame.mixer.music.pause()
        clock.tick(10)
        screen.blit(game_over_bg, (0, 0))
        pygame.display.update()


# Main game code
class MAIN:

    def __init__(self):
        self.play_background_music()
        self.snake = SNAKE()
        self.fruit = FRUIT()

    # Update all
    def update(self):
```

```
            self.snake.move_snake()
            self.check_collision()
            self.check_fail()

        # Draw elements
        def draw_elements(self):
            self.draw_grass()
            self.fruit.draw_fruit()
            self.snake.draw_snake()
            self.draw_score()

        # Check if snake collide the fruit
        def check_collision(self):
            if self.fruit.pos == self.snake.body[0]:
                self.fruit.randomize()
                self.snake.add_block()
                self.snake.play_crunch_sound()

            for block in self.snake.body[1:]:
                if block == self.fruit.pos:
                    self.fruit.randomize()

        # Check if snake collide wall and itself and then game over
        def check_fail(self):
            if not 0 <= self.snake.body[0].x < cell_number or not 0 <=
self.snake.body[0].y < cell_number:
                self.play_wall_crash_sound()
                self.game_over()
                Game_Over()
                pygame.mixer.music.unpause()

            for block in self.snake.body[1:]:
                if block == self.snake.body[0]:
                    self.game_over()

        # Reseting the snake after game over
            def game_over(self):
            self.snake.reset()
```