```
import random
import tkinter as tk

win1=tk.Tk()

def Game_of_life():
    win1.destroy()
    class GameOfLife:
        def __init__(self, rows, columns, cell_size=10, speed=1):
            self.rows = rows
            self.columns = columns
            self.cell_size = cell_size
            self.speed = speed
            self.window = tk.Tk()
            self.window.title("Game of Life")
            self.window.resizable(True,False)
            self.canvas = tk.Canvas(self.window, width=self.columns * self.cell_size, height=self.rows * self.ce
ll_size, bg='white')
            self.canvas.pack()
            self.grid = self.create_grid()
            self.create_buttons()
            self.create_label()
            self.draw_grid()
            self.window.mainloop()

        def create_buttons(self):
            self.start_button = tk.Button(self.window, text='Start',fg='black',bg='green', command=self.run)
            self.reset_button = tk.Button(self.window, text='Reset',fg='black',bg='yellow', command=self.reset)

            self.stop_button = tk.Button(self.window, text='Exit',fg='black',bg='red',command=self.window.dest
roy)
            self.num_gen_label = tk.Label(self.window, text='Number of Generations:',fg='black',font=('Elepha
nt',10))
            self.num_gen_entry = tk.Entry(self.window)

            self.speed_slider=tk.Scale(self.window,from_=1,to=100,orient=tk.HORIZONTAL,label='speed',font
=('Elephant',10),command=self.change_speed)
            self.speed_slider.set(self.speed)

            self.start_button.pack()
            self.reset_button.pack()
            self.stop_button.pack()


            self.num_gen_label.pack()
            self.num_gen_entry.pack()

            self.speed_slider.pack()

        def create_label(self):
            self.generation = 0
            self.label = tk.Label(self.window, text=f'Generation:{self.generation}')
            self.label.pack()

        def update_generation(self):
```

```
        self.label.config(text=f'Generation:{self.generation}')

    def create_grid(self):
        grid = []
        for row in range(self.rows):
            grid.append([])
            for col in range(self.columns):
                grid[row].append(random.randint(0, 1))
        return grid

    def draw_grid(self):
        for row in range(self.rows):
            for column in range(self.columns):
                x1 = column * self.cell_size
                y1 = row * self.cell_size
                x2 = x1 + self.cell_size
                y2 = y1 + self.cell_size
                if self.grid[row][column] == 1:
                    self.canvas.create_rectangle(x1, y1, x2, y2, fill='violet', outline='black')
                else:
                    self.canvas.create_rectangle(x1, y1, x2, y2, fill='brown', outline='black')

    def update(self):
        if self.remaining_generations > 0:
            new_grid = []
            for row in range(self.rows):
                new_grid.append([])
                for column in range(self.columns):
                    new_grid[row].append(self.get_new_cell_state(row, column))
            self.grid = new_grid
            self.canvas.delete('all')
            self.draw_grid()
            self.window.after(500 // self.speed, self.update)
            self.generation += 1
            self.remaining_generations -= 1
            self.update_generation()

    def get_new_cell_state(self, row, column):
        num_neighbours = self.get_num_neighbours(row, column)
        if self.grid[row][column] == 1:
            if num_neighbours < 2:
                return 0
            elif num_neighbours == 2 or num_neighbours == 3:
                return 1
            elif num_neighbours > 3:
                return 0
        else:
            if num_neighbours == 3:
                return 1
            else:
                return 0

    def get_num_neighbours(self, row, column):
        num_neighbours = 0
        for i in range(-1, 2):
```

```
            for j in range(-1, 2):
                if i == 0 and j == 0:
                    continue
                neighbour_row = row + i
                neighbour_column = column + j
                if neighbour_row < 0 or neighbour_row >= self.rows or neighbour_column < 0 or neighbour_c
olumn >= self.columns:
                    continue
                num_neighbours += self.grid[neighbour_row][neighbour_column]
        return num_neighbours

    def run(self):
        num_gen_input = self.num_gen_entry.get()
        if num_gen_input.isdigit():
            self.remaining_generations = int(num_gen_input)
            self.update()

    def reset(self):
        self.grid = self.create_grid()
        self.canvas.delete('all')
        self.draw_grid()
        self.generation = 0
        self.update_generation()

    def change_speed(self, speed):
        self.speed = int(speed)

if __name__ == '__main__':
  game = GameOfLife(50, 50, 10, 100)

image1=tk.PhotoImage(file='start.png')
label1=tk.Label(win1,image=image1)
label1.pack()
button1=tk.Button(win1,text='PLAY',bg='blue',font=('Eras Bold ITC',20),height=1,width=8,command=lamb
da:Game_of_life())
button1.place(x=500,y=380)
win1.resizable(False,False)
win1.mainloop()
'''if __name__ == '__main__':
    game = GameOfLife(25, 25, 10, 100)'''
```