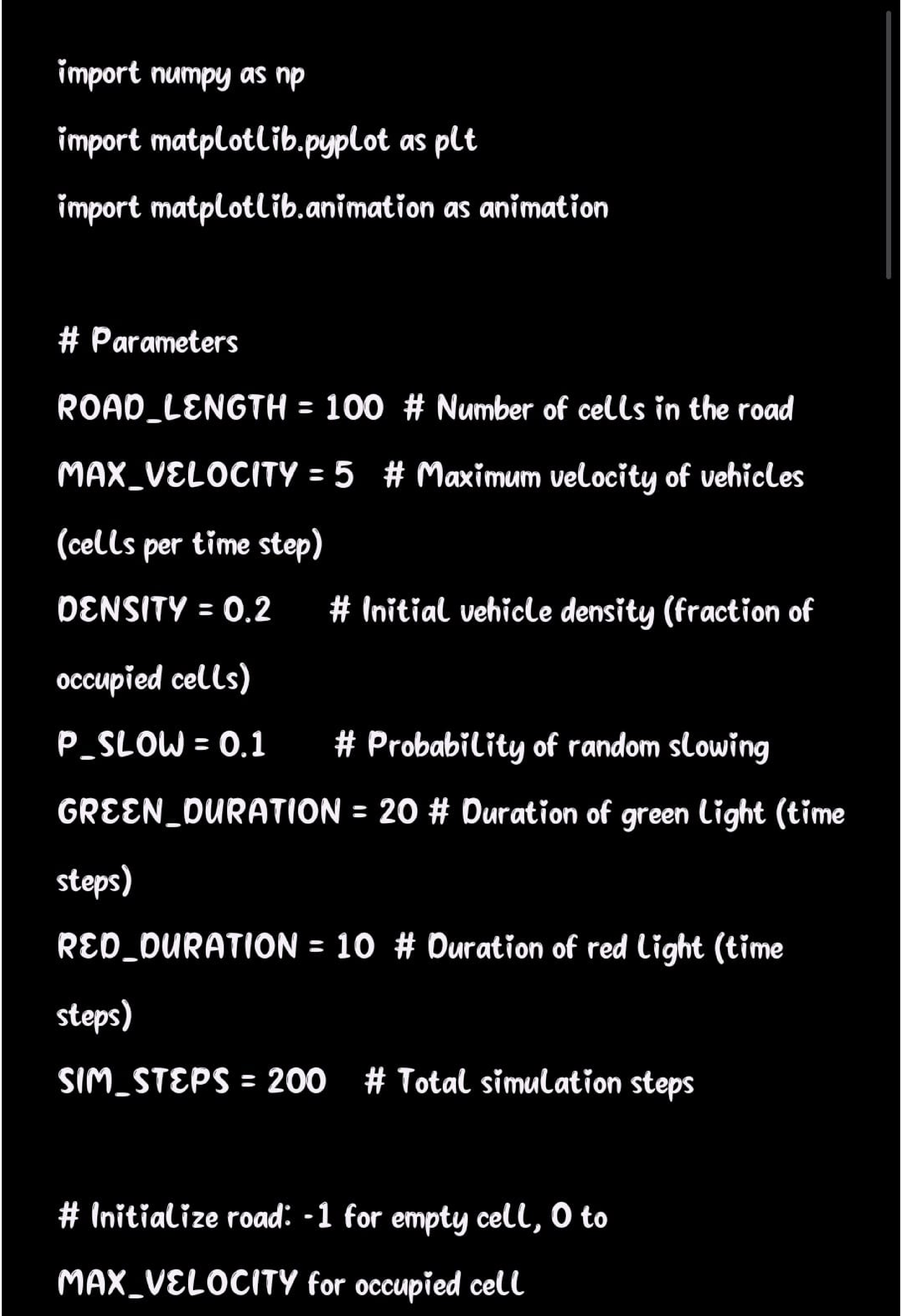
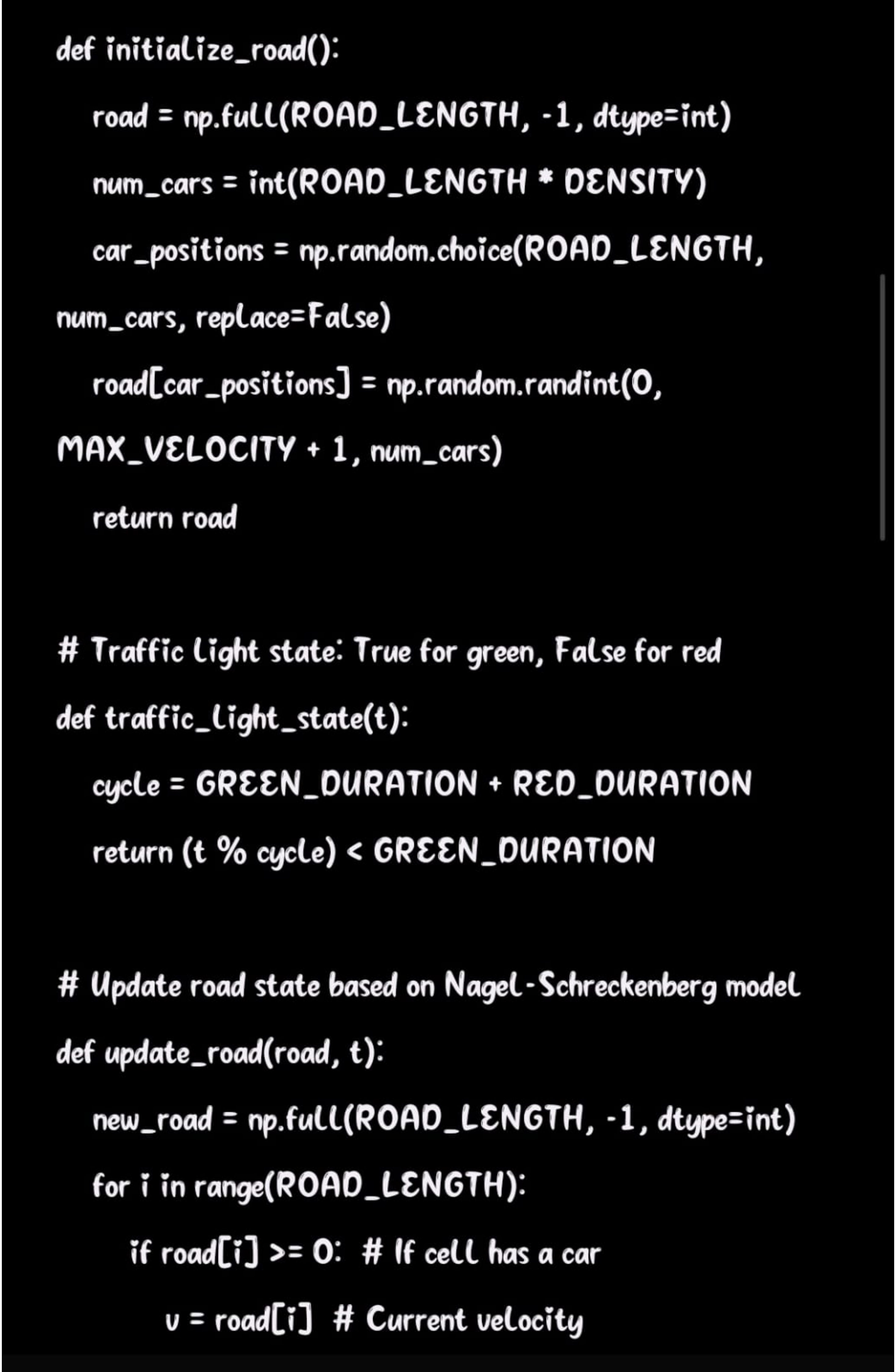
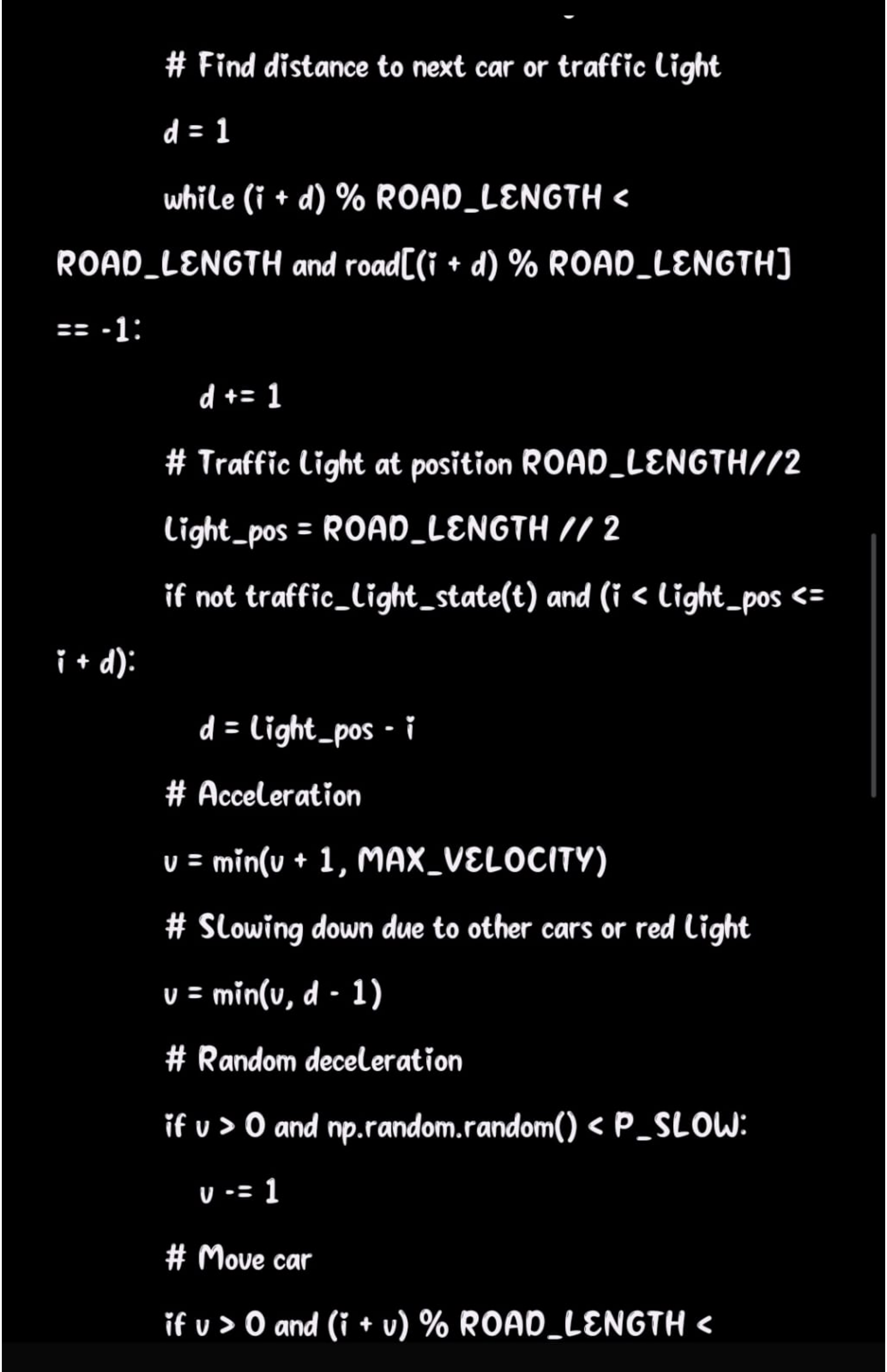


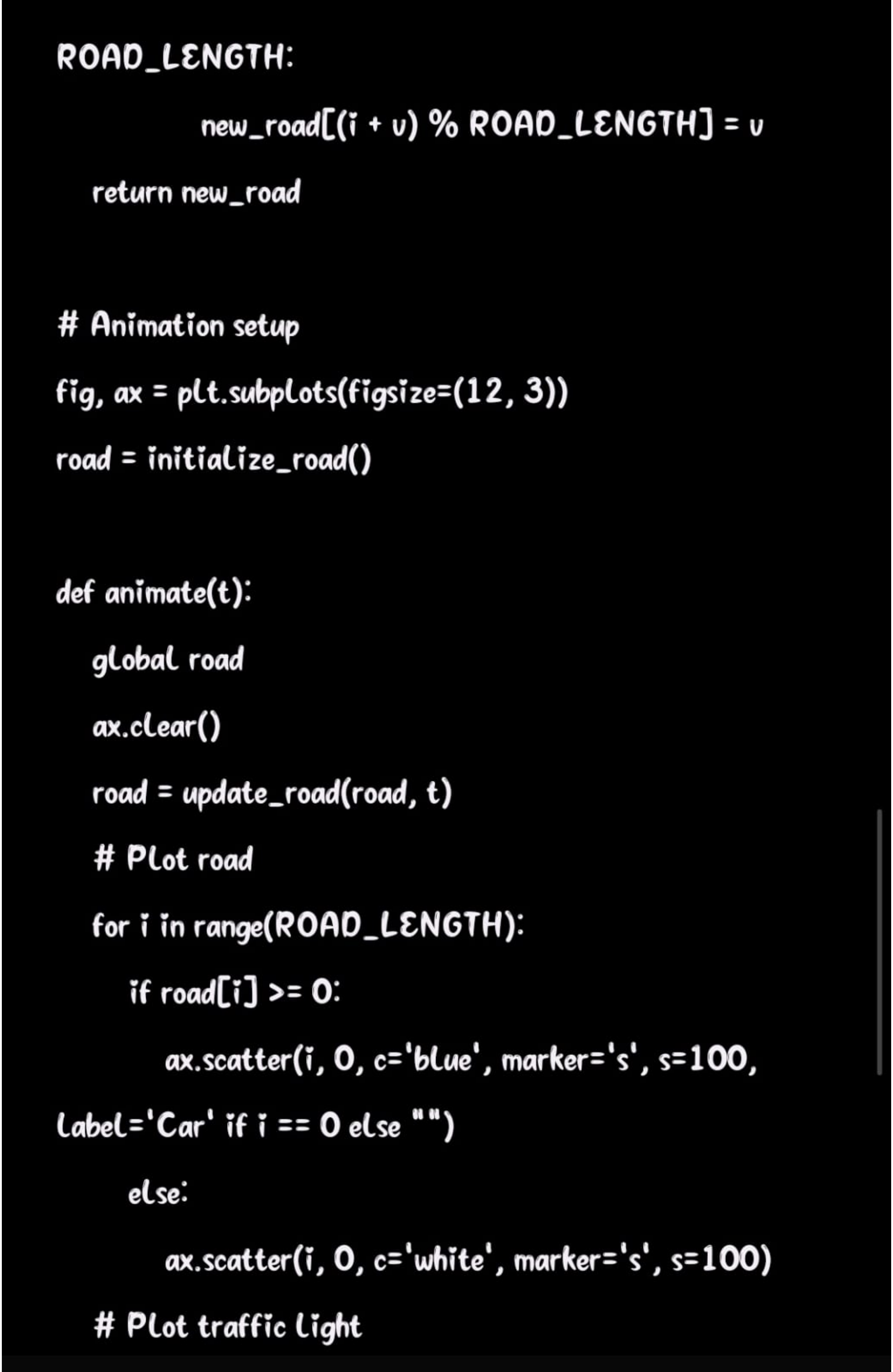
```
import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation


# Parameters

ROAD_LENGTH = 100  # Number of cells in the road

MAX_VELOCITY = 5   # Maximum velocity of vehicles

(cells per time step)

DENSITY = 0.2      # Initial vehicle density (fraction of

occupied cells)

P_SLOW = 0.1       # Probability of random slowing

GREEN_DURATION = 20 # Duration of green light (time

steps)

RED_DURATION = 10  # Duration of red light (time

steps)

SIM_STEPS = 200    # Total simulation steps


# Initialize road: -1 for empty cell, 0 to

MAX_VELOCITY for occupied cell
```



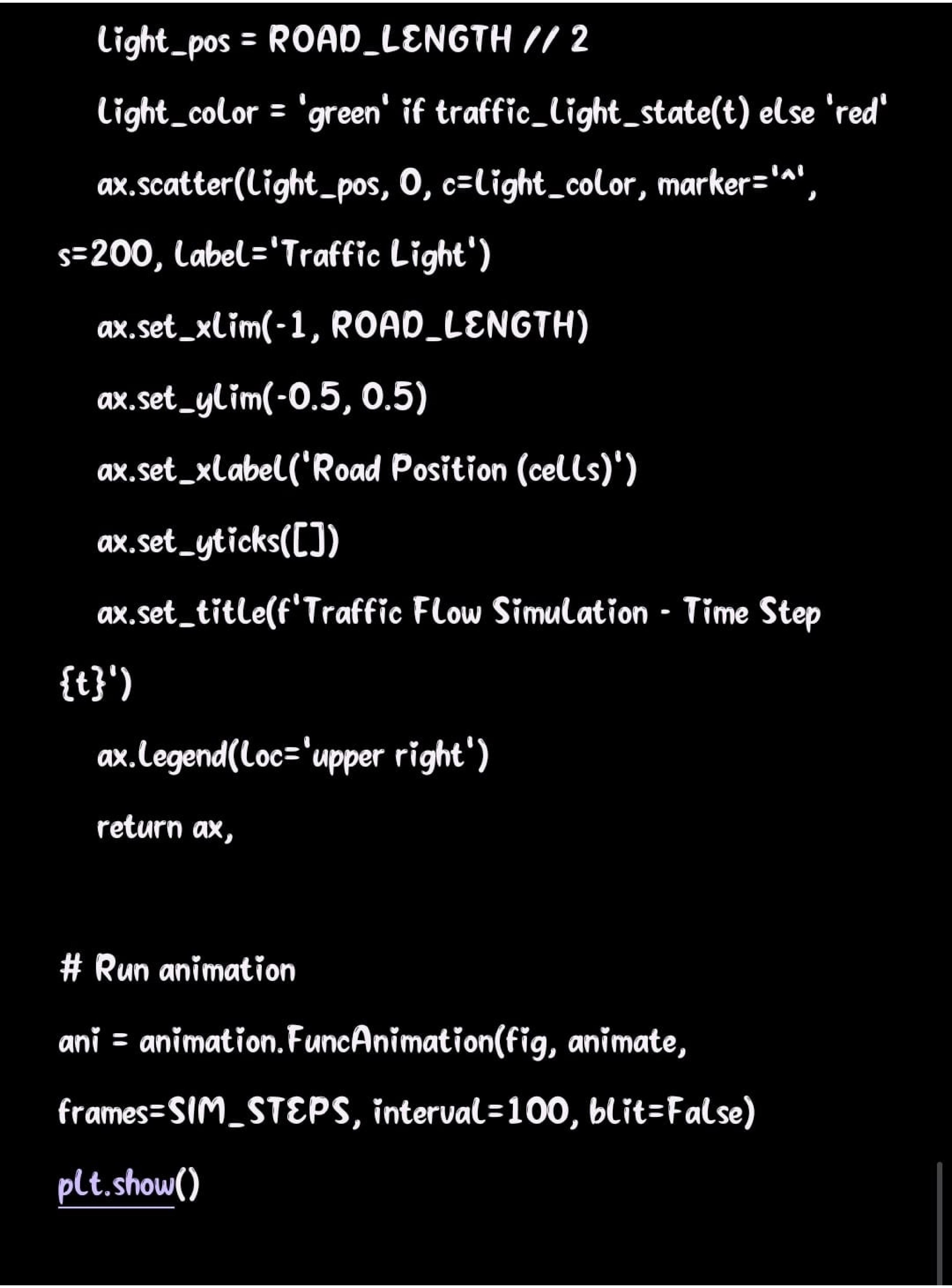
```
def initialize_road():
    road = np.full(ROAD_LENGTH, -1, dtype=int)
    num_cars = int(ROAD_LENGTH * DENSITY)
    car_positions = np.random.choice(ROAD_LENGTH,
num_cars, replace=False)
    road[car_positions] = np.random.randint(0,
MAX_VELOCITY + 1, num_cars)
    return road

# Traffic light state: True for green, False for red
def traffic_light_state(t):
    cycle = GREEN_DURATION + RED_DURATION
    return (t % cycle) < GREEN_DURATION

# Update road state based on Nagel-Schreckenberg model
def update_road(road, t):
    new_road = np.full(ROAD_LENGTH, -1, dtype=int)
    for i in range(ROAD_LENGTH):
        if road[i] >= 0:  # If cell has a car
            v = road[i]  # Current velocity
```



```
        # Find distance to next car or traffic light
        d = 1
        while (i + d) % ROAD_LENGTH < ROAD_LENGTH and road[(i + d) % ROAD_LENGTH] == -1:
            d += 1
        # Traffic light at position ROAD_LENGTH//2
        light_pos = ROAD_LENGTH // 2
        if not traffic_light_state(t) and (i < light_pos <= i + d):
            d = light_pos - i
        # Acceleration
        v = min(v + 1, MAX_VELOCITY)
        # Slowing down due to other cars or red light
        v = min(v, d - 1)
        # Random deceleration
        if v > 0 and np.random.random() < P_SLOW:
            v -= 1
        # Move car
        if v > 0 and (i + v) % ROAD_LENGTH <
```



```
ROAD_LENGTH:
                new_road[(i + v) % ROAD_LENGTH] = v
    return new_road


# Animation setup
fig, ax = plt.subplots(figsize=(12, 3))
road = initialize_road()


def animate(t):
    global road
    ax.clear()
    road = update_road(road, t)
    # Plot road
    for i in range(ROAD_LENGTH):
        if road[i] >= 0:
            ax.scatter(i, 0, c='blue', marker='s', s=100,
label='Car' if i == 0 else "")
        else:
            ax.scatter(i, 0, c='white', marker='s', s=100)
    # Plot traffic light
```



```python
    light_pos = ROAD_LENGTH // 2
    light_color = 'green' if traffic_light_state(t) else 'red'
    ax.scatter(light_pos, 0, c=light_color, marker='^', s=200, label='Traffic Light')
    ax.set_xlim(-1, ROAD_LENGTH)
    ax.set_ylim(-0.5, 0.5)
    ax.set_xlabel('Road Position (cells)')
    ax.set_yticks([])
    ax.set_title(f'Traffic Flow Simulation - Time Step {t}')
    ax.legend(loc='upper right')
    return ax,

# Run animation
ani = animation.FuncAnimation(fig, animate, frames=SIM_STEPS, interval=100, blit=False)
plt.show()
```