Design Document

1. Description of Code and program mechanism

The code simulates a simple traffic light system that includes a pedestrian crossing. The simulation runs for a total duration of 200 seconds and is divided into different time intervals. The system has three possible states: "red", "yellow", and "green". When the system is in the "red" state, cars are expected to stop, and pedestrians are not allowed to cross. In the "yellow" state, the light is about to turn red, and pedestrians should finish crossing if they have already started. In the "green" state, cars are allowed to proceed, and pedestrians can cross the street.

The code begins by initializing several variables, including a counter variable count that keeps track of the time elapsed, system_state that tracks the current state of the system, duration that represents the cycle duration of the traffic light, yellow_duration that represents the duration of the yellow state, timestep that represents the time increment used in the simulation, and pedestrian_moment that represents the time at which a pedestrian presses the crossing button.

The code then enters a while loop that iterates over each time step from 0 to 200 seconds. Within the loop, the code uses a series of conditional statements to determine the current state of the system and the appropriate system output and input. The code also prints out the current system state and any output or input generated at each time step.

The first conditional statement checks if the current time is less than or equal to the traffic light cycle duration, and if so, sets the system state to "red". The second conditional statement checks if the current time is between the traffic light cycle duration and twice the duration, and if so, generates a "sigG" output if the pedestrian has not pressed the crossing button or sets the system state to "pending" if the pedestrian has pressed the button. The third conditional statement checks if the current time is between twice the duration and the duration plus the yellow duration, and if so, generates a "sigY" output and sets the system state to "yellow". The fourth conditional statement checks if the current time is between the duration plus the yellow duration and three times the duration, and if so, generates a "sigR" output and sets the system state to "red". The final conditional statement checks if the current time is greater than three times the duration and generates a "sigG" output and sets the system state to "green".

In addition to tracking the system state, the code also tracks the input variable, which represents any user input (i.e., the pedestrian pressing the crossing button) and the output variable, which represents the traffic light signal (i.e., "sigR", "sigY", or "sigG"). The code prints out these variables at the appropriate time steps to reflect the current state of the system.

Overall, the given Python code is a simple but effective simulation of a traffic light system with pedestrian crossing. It could benefit from some refactoring to make it more readable and easier to modify, but it provides a good starting point for more complex simulations.

2. Screenshots of the program and output

The expected output of the code can be described based on the printed statements. The code simulates a traffic light system with a duration of 60 seconds for each state (green, yellow, and red), with a total simulation time of 200 seconds.

The expected output of the code will print the current time, the current system state, and any corresponding system input or output. The simulation starts with the system state set to "red" for the first 60 seconds. The state then transitions to "green" for the next 50 seconds until a pedestrian arrives and presses the pedestrian button at 70 seconds. The system then transitions to a "pending" state, waiting for the pedestrian to cross, for 10 seconds. After the 10 seconds are up, the system transitions to a "yellow" state for 5 seconds and then to a "red" state for the next 60 seconds. Finally, the system transitions back to a "green" state for the last 70 seconds of the simulation.

The output should resemble the following:

Time: 0 Seconds

System State: red

Time: 1 Seconds

System State: red

Time: 2 Seconds

System State: red

...

Time: 59 Seconds

System State: red

Output: sigG

System State: green

...

Time: 69 Seconds

Input: pedestrian

System State: pending

...

Time: 75 Seconds

Output: sigY

System State: yellow

...

Time: 119 Seconds

Output: sigY

System State: yellow

...

Time: 120 Seconds

Output: sigR

System State: red

...

Time: 179 Seconds

Output: sigR

System State: red

...

Time: 180 Seconds

Output: sigG

System State: green

...

Time: 199 Seconds

Output: sigG

System State: green