Queue Simulation Code - Simple Explanation

What This Code Does

This code simulates a simple queue system, like what you'd see at a bank or coffee shop. It tracks customers coming in, waiting in line, getting served, and leaving.

The Main Parts

MM1Queue Class

This is the heart of the simulation. Think of it as a virtual service counter.

What it tracks:

- How fast customers arrive (lambda_rate)
- How fast the server works (mu_rate)
- How long to run the simulation
- Whether the server is busy or free
- How many people are waiting in line

Key Functions Explained

`__init__` - Sets up the simulation

- Like setting up a new store before opening
- Decides how busy it will be and for how long

`exp_time` - Calculates random waiting times

- Uses math to make realistic random delays
- Some customers arrive quickly, others take longer

`update_state` - Keeps track of changes

- Like a stopwatch that records how long each situation lasts
- Tracks when the line gets longer or shorter

`run` - The main simulation loop

- This is where all the action happens
- Customers arrive, wait, get served, and leave
- Keeps going until the time runs out

`get_metrics` - Calculates the final results

- Like looking at the day's receipts
- Shows averages, busy times, and customer satisfaction

The Four Test Scenarios

The code tests four different situations:

1. **Original (ρ = 0.33)** - Pretty quiet, server relaxed
2. **Scenario 1 (ρ = 0.5)** - Moderate busy, balanced
3. **Scenario 2 (ρ = 0.9)** - Very busy, server stressed
4. **Scenario 3 (ρ = 0.1)** - Super quiet, server mostly free

What Gets Measured

Customer Stats:

- How many got served
- Average waiting time
- Average time spent in the store

Server Stats:

- How busy the server was
- How much free time they had
- When they were overwhelmed

Queue Stats:

- Average line length
- How often the line was empty
- Peak busy periods

The Results

The code compares what should happen in theory versus what actually happens in the simulation. It shows this with:

- Line graphs showing trends
- Bar charts comparing scenarios
- Detailed logs of individual customers

Why This Matters

This type of simulation helps businesses:

- Decide how many servers to hire
- Predict busy periods
- Improve customer experience
- Save money on staffing

The Math Behind It (Simple Version)

ρ (rho) = arrival rate ÷ service rate

- If $\rho < 1$: System can handle the load
- If $\rho > 1$: System gets overwhelmed
- If $\rho = 0.9$: Server is busy 90% of the time

Key Takeaways

- Higher arrival rates = longer lines
- Faster service = happier customers
- Balance is everything in queue management
- Real results usually match theoretical predictions

The code proves that math can predict real-world behavior pretty accurately!