Clinic Simulation

Waiting time Analysis and Patient Management

Content Outline

Topics for discussion

Introduction

Simulating the daily operation of a doctor's office from 12 noon to 4 pm.

- The aim of this simulation is to simulate the experience of patients inside the office from the moment they arrive until the end of their visit to the doctor.

- Medical clinics are places that require precise management to organize patient appointments, handle waiting times, and ensure effective service delivery. Therefore, we designed a simulation program in the Python programming language that reflects the real operations in a doctor's office. The program depends on several factors, such as the number of patients arriving at the office per hour, the various ages of patients, and the time each patient takes to visit the doctor, which is calculated according to the patient's age.

The main problem

How to simulate a doctor's office and determine patient waiting times.

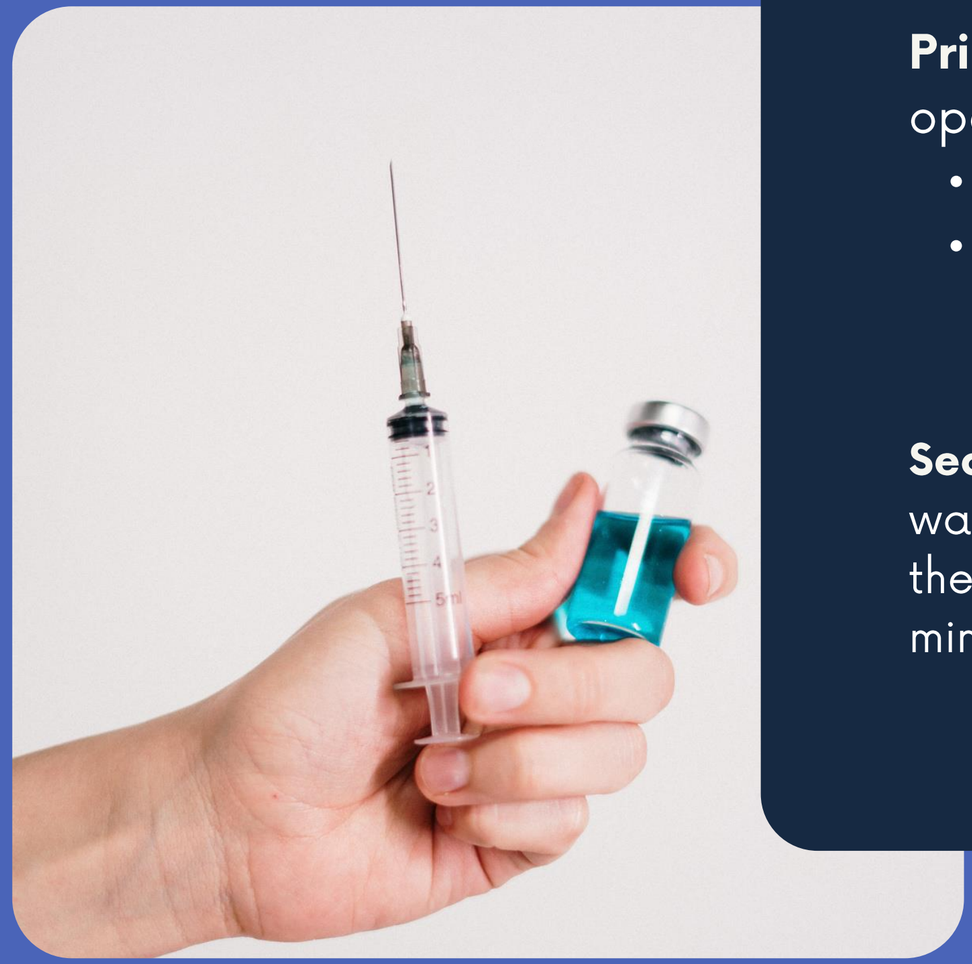
Key Points

Problem Overview

- We need to simulate the operation of a doctor's clinic from 12 pm to 4 pm.
- On an average day, approximately 10 patients arrive per hour.
- Each patient's age is randomly chosen between 20 and 60, with equal probability for any age.
- Patients are treated in a first-come, first-served manner.
- The time each patient spends with the doctor is calculated using the formula: Patient time with doctor = Age / 5 minutes.
- We are also asked to consider the impact if the formula changes to Patient time with doctor = Age / 10 minutes.

Key Points

Objective



Primary Objective: Simulate the clinic's operation to calculate:

- The average waiting time of patients.
- The number of patients still waiting at 4 pm.

Secondary Objective: Recalculate the average waiting time and the number of remaining patients if the patient treatment time is changed to Age / 10 minutes.

Program Structure

1. Program Structure Overview

2. Detailed Breakdown of Each Class and Function

A. Queue Class

B. Patients Class

C. Clinic Class

D. Clinic_Simulation Function

1. Program Structure Overview

The program simulates the operation of a doctor's clinic from 12 PM to 4 PM, using Python classes to model different aspects of the clinic's operations. The main components of the code are:

- **Queue Class:** Represents the queue of patients waiting to see the doctor.
- **Patients Class:** Models individual patients, including their arrival time and age.
- **Clinic Class:** Manages the doctor's availability and service times.
- **Clinic_Simulation Function:** The main function that runs the simulation, simulates patient arrivals, processes the queue, and calculates average waiting times.

2. Detailed Breakdown of Each Class and Function

A. Queue Class

Purpose: To manage the queue of patients waiting to be seen by the doctor.

Methods:

__init__(): Initializes an empty list to hold the patients.
is_empty(): Checks if the queue is empty.
enqueue(item): Adds a new patient to the queue.
dequeue(): Removes and returns the first patient in the queue.
size(): Returns the current size of the queue.

A. Queue Class

```python
# Queue Class: Represents the queue of patients waiting to see the doctor
class Queue:
    def __init__(self):
        # Initializes an empty queue
        self.items = []

    def is_empty(self):
        # Checks if the queue is empty
        return len(self.items) == 0

    def enqueue(self, item):
        # Adds an item (patient) to the end of the queue
        self.items.append(item)

    def dequeue(self):
        # Removes and returns the item (patient) from the front of the queu
        return self.items.pop(0) if not self.is_empty() else None

    def size(self):
        # Returns the number of items in the queue
        return len(self.items)
```

B. Patients Class

Purpose: **Represents a patient and includes properties such as age and arrival time.**

Methods:

__init__(arrival_time):

Initializes a patient with a random age between 20 and 60 and records the arrival time.

wait_time(current_time):

Calculates the waiting time based on the current time and the arrival time.

get_age():

Returns the age of the patient.

B. Patients Class

```python
import random #inserting the library to use "randrange"

class Patient:
    def __init__(self, arrival_time):
        self.age = random.randrange(20, 61)
        self.arrival_time = arrival_time

    def get_age(self):
        return self.age

    def wait_time(self, current_time):
        """
        Compute the amount of time the patient
        waited in the queue before entering
        """
        return current_time - self.arrival_time
```

C. Clinic Class

Purpose: **Manages the doctor’s current patient and the time remaining for their treatment.**

Method:

__init__(rate):

Initializes the clinic with a patient rate (time per age unit).

tick()

Decreases the current patient's remaining time by 1 minute, indicating that a minute has passed.

is_busy

Checks if the doctor is currently busy.

start_service

Assigns a new patient to the doctor and sets their treatment time.

```python
class Clinic:
    def __init__(self, service_rate):
        self.service_rate = service_rate
        self.current_patient = None
        self.remaining_time = 0

    def is_busy(self):
        return self.current_patient is not None

    def start_service(self, patient):
        """
        Simulate the entering
        of the patients to the doctor
        """
        self.current_patient = patient
        self.remaining_time = ( patient.get_age() / self.service_rate ) * 60

    def tick(self):
        """
        Simulate the clock in the clinic
        one call for this methode means that 1 minute has passed
        """
        if self.current_patient:
            self.remaining_time -= 1
            if self.remaining_time <= 0:
                self.current_patient = None
```

C. Clinic Class

D. simulate Function

Purpose: **Runs the simulation for a specified period, simulates patient arrivals, manages the queue, and calculates waiting times.**

Parameters

total_minutes

Total time (in minutes) for the simulation.

service_rate

Rate for calculating the time each patient spends with the doctor.

Logic:

Simulates patient arrivals using a random condition

Dequeues patients and processes them if the doctor is not busy.

Tracks and calculates average waiting times.

Prints the final results after the simulation ends.

Explanation of the Program Flow:

Simulation Start

The Simulate function is executed for a fixed time span (4 hours or 240 minutes).

Patient Arrival:

A random chance (1/6 per minute) determines if a new patient arrives and is added to the queue.

Processing Patients:

- If the doctor is free, the next patient in the queue is dequeued and starts treatment.
- The treatment time for each patient is calculated based on the patient's age and the given rate.

Time Tracking:

The tick() method in the Clinic class reduces the time left for the current patient.

Output:

At the end of the simulation, the average waiting time and the number of patients remaining in the queue are printed.

D. simulate Function
Part 2 (Simulate())

```
import random
from Queue import *
from Clinic import *
from Patient import *


def simulate(total_hours, service_rate):
    total_seconds = total_hours * 60 * 60
    clinic = Clinic(service_rate)
    patient_queue = Queue()
    waiting_times = []

    for current_minute in range(total_seconds):
        # Check if a new patient arrives (1 in 361 chance per seconds)
        if random.randrange(1,361) == 77:
            new_patient = Patient(current_minute)
            patient_queue.enqueue(new_patient)

        # Check if the patient can see the doctor or not
        if not clinic.is_busy() and not patient_queue.is_empty():
            next_patient = patient_queue.dequeue()
            clinic.start_service(next_patient)
            waiting_times.append(next_patient.wait_time(current_minute))

        clinic.tick()

    # Calculate average waiting time
    average_wait = sum(waiting_times) / len(waiting_times) / 60 if waiting_times else 0
    print(f"Average Wait: {average_wait:.2f} minutes | Patients Remaining: {patient_queue.size()}")
```

D. simulate Function

Part 2 (Main())

```
if __name__ == "__main__":
    print("Simulation for Service Time = Age / 5:")
    for _ in range(3):
        simulate(4, 5)

    print("\n" + "#" * 40 + "\n")

    print("Simulation for Service Time = Age / 10:")
    for _ in range(3):
        simulate(4, 10)
```

Possible scenarios

1. Smooth Operation (No Overcrowding)

2. High Patient Arrival Rate (Overcrowding)

3. Low Patient Arrival Rate (Underutilized Doctor)

4. Young Patients Majority

5. Older Patients Majority

6. Mixed Service Rates (Comparison Scenario)

7. Random Surge Towards End of Simulation

8. Unlucky Randomness

1. Smooth Operation (No Overcrowding)

Description: Patients arrive at a manageable rate, and the doctor efficiently handles the queue without significant delays.

Outcome:

- The queue rarely grows long.
- Average waiting time is minimal.
- Most patients are seen by 4 PM, leaving few or no remaining patients.

2. High Patient Arrival Rate (Overcrowding)

Description: Patients arrive more frequently than usual (e.g., several consecutive arrivals due to randomness), causing the queue to grow rapidly.

Outcome:

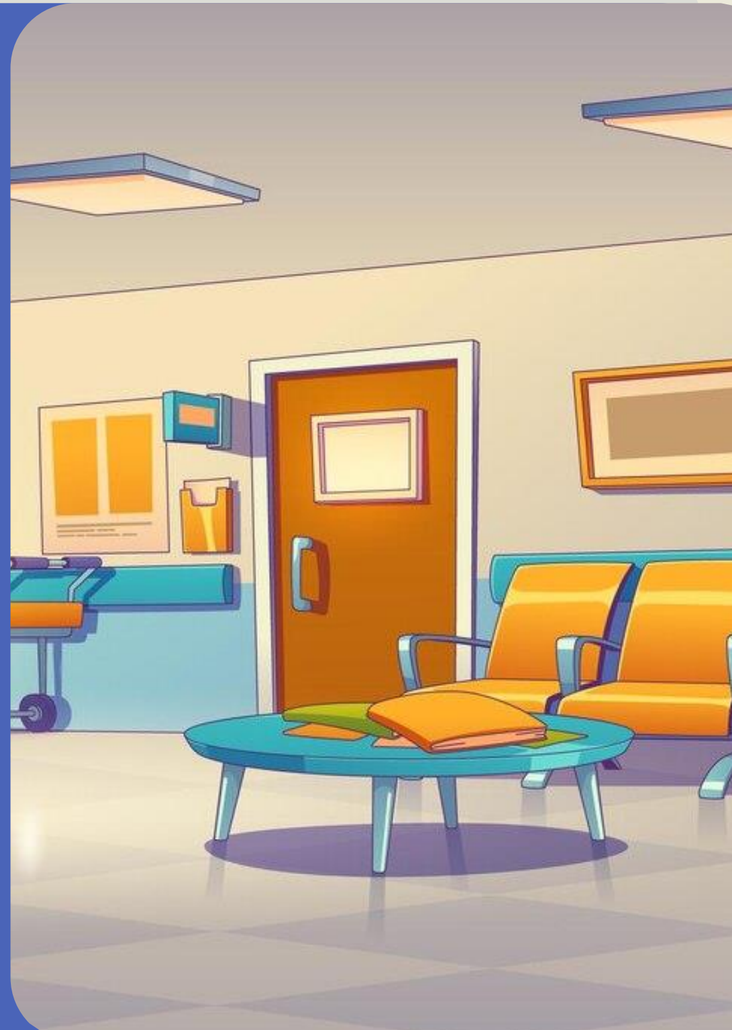
- The queue lengthens, and waiting times increase significantly.
- Many patients remain untreated by 4 PM due to limited doctor availability.
- Average waiting time rises sharply.

3. Low Patient Arrival Rate (Underutilized Doctor)

Description: Fewer patients arrive than expected, leaving the doctor idle for extended periods.

Outcome:

- The queue remains empty for much of the simulation.
- Average waiting time is extremely low or even zero.
- All patients are served before the simulation ends, and the clinic operates below capacity.



4. Young Patients Majority

Description: Randomly, the majority of patients have lower ages (closer to 20).

Outcome:

- Patient service times are shorter due to the (Age/5) formula.
- The queue moves quickly, and most or all patients are treated.
- Average waiting time is low.

5. Older Patients Majority

Description: Randomly, the majority of patients are older (closer to 60).

Outcome:

- Patient service times are longer due to the (Age/5) formula.
- The queue grows longer, and fewer patients are served within the simulation period.
- Average waiting time is higher, and more patients remain untreated by 4 PM.

6. Mixed Service Rates (Comparison Scenario)

Description: Run simulations for both (Age/5) and (Age/10) service rates.

Outcome:

- For (Age/5): Higher average waiting times but potentially fewer remaining patients since treatment is slower.
- For (Age/10): Lower average waiting times but possibly more patients remaining untreated as treatment is faster but capacity might be insufficient for high arrivals.

7. Random Surge Towards End of Simulation

Description: A sudden surge in patient arrivals occurs closer to 4 PM.

Outcome:

- The queue grows significantly, leaving many patients untreated.
- The average waiting time depends on when patients arrive; late arrivals might not even start service.

8. Unlucky Randomness

Description: A large number of patients arrive at almost the same time early in the simulation.

Outcome:

- Early arrivals increase the initial queue length.
- The clinic struggles to recover from the backlog, causing high waiting times throughout the simulation.
- Many patients remain untreated.

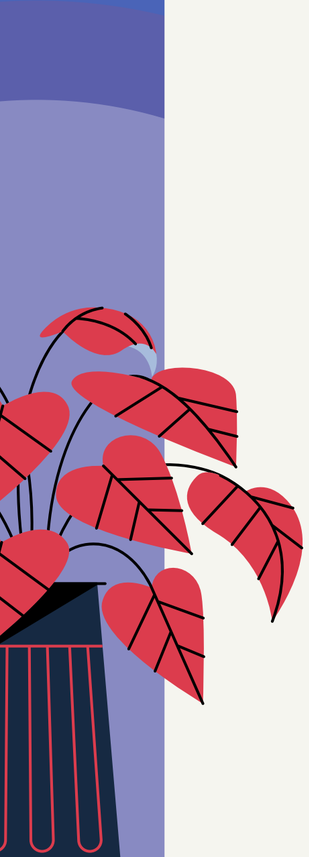
Simulation results

To determine the simulation result, we can run the program with the provided parameters and analyze its output. Since the simulation is stochastic (randomized), results will vary for each run. However, we can describe the expected results and what to look for.

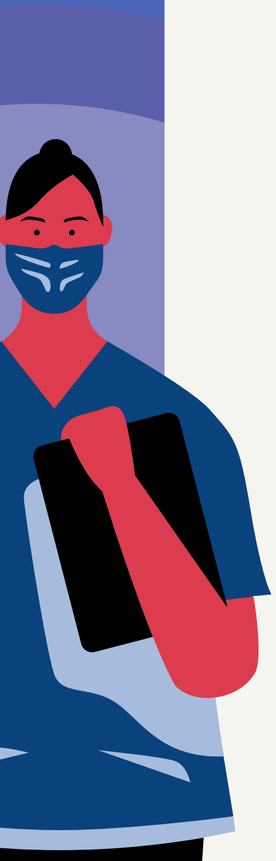
Expected Results:

The program will output:

1. Average Waiting Time: How long patients, on average, waited in the queue before being treated by the doctor.
2. Remaining Patients: The number of patients still waiting in the queue at the end of the simulation (4 PM).
3. Total Patients: The total number of patients who arrived at the clinic during the simulation period.



Simulation results.....



Key Observations from Results:

If the service rate is (Age/5):

Average waiting times will generally be higher due to longer patient consultation times.

More patients may remain in the queue at the end of the simulation if patient arrivals exceed the doctor's capacity.

If the service rate is (Age/10):

Average waiting times will be lower since the doctor spends less time with each patient.

Fewer patients are likely to remain in the queue by 4 PM.

OUTPUT EXAMPLE

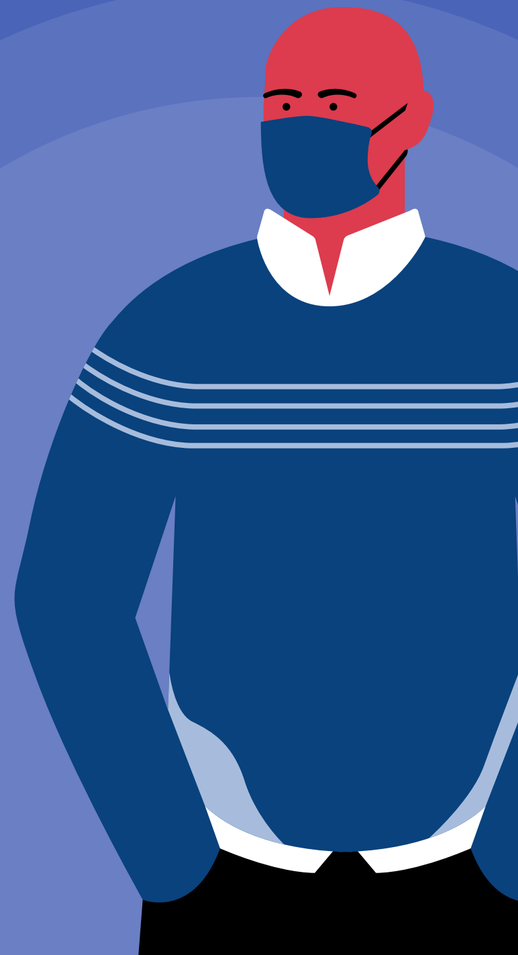
WHEN YOU RUN THIS CODE, THE OUTPUT WILL LOOK SOMETHING LIKE THIS:

(BUT NOT THIS)

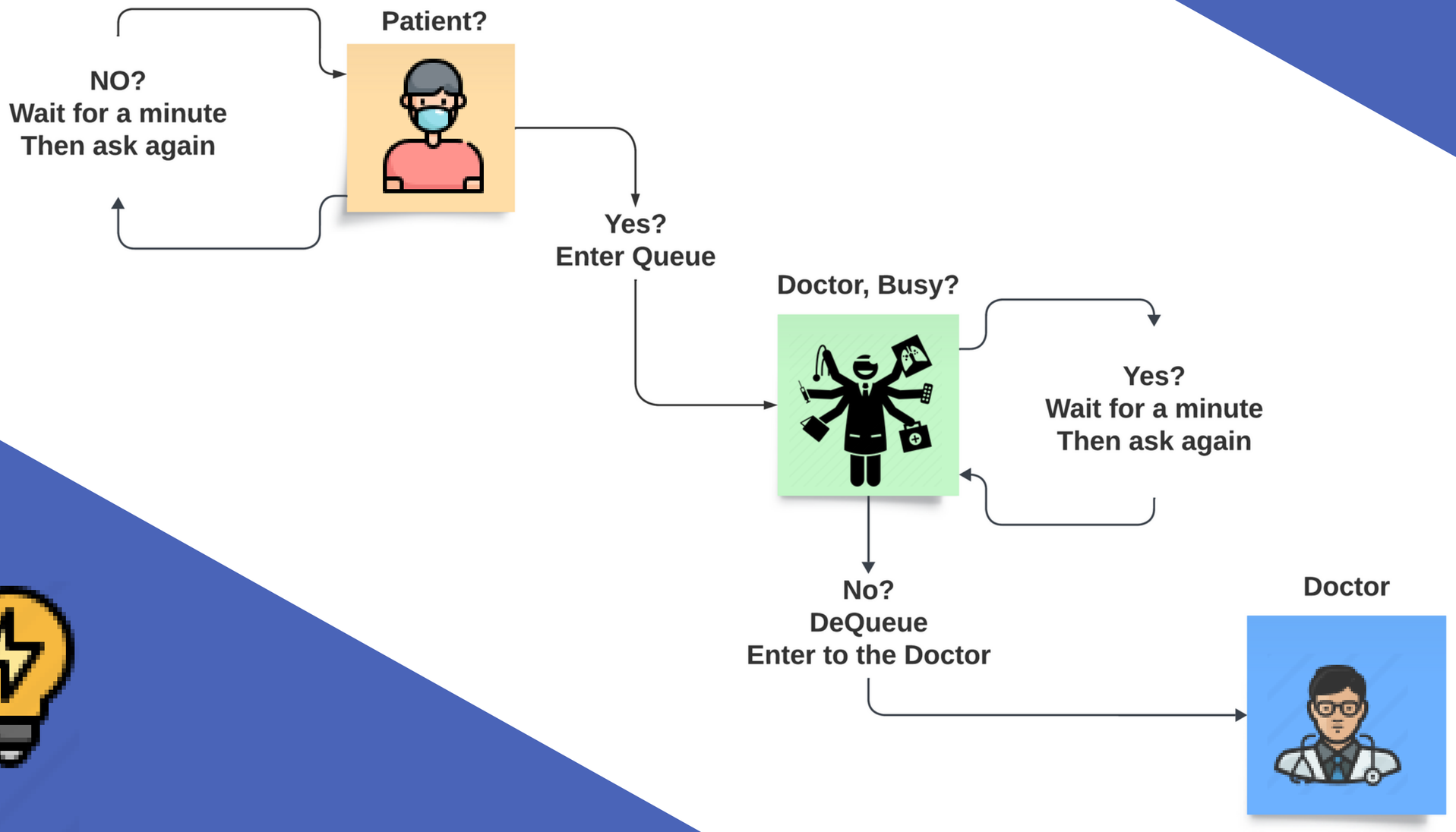
```
Simulation for Service Time = Age / 5:
Average Wait: 15.34 minutes | Patients Remaining: 5
Average Wait: 12.21 minutes | Patients Remaining: 3
Average Wait: 14.78 minutes | Patients Remaining: 7

##################################################

Simulation for Service Time = Age / 10:
Average Wait: 7.89 minutes | Patients Remaining: 2
Average Wait: 6.43 minutes | Patients Remaining: 1
Average Wait: 5.67 minutes | Patients Remaining: 0
```

CONCLUSION:

- THE PROGRAM SIMULATES A CLINIC'S OPERATIONS, SHOWCASING PATIENT MANAGEMENT AND DOCTOR INTERACTIONS USING OBJECT-ORIENTED PRINCIPLES.
- IT EVALUATES CLINIC EFFICIENCY THROUGH METRICS LIKE AVERAGE WAITING TIME AND REMAINING PATIENTS, INFLUENCED BY SERVICE RATES.
- THE SIMULATION HIGHLIGHTS HOW SHORTER SERVICE TIMES IMPROVE THROUGHPUT, MAKING IT A PRACTICAL TOOL FOR STUDYING CLINIC PERFORMANCE AND OPTIMIZING OPERATIONS.



CONCLUSION:


Patient?
NO?
Wait for a minute
Then ask again
Yes?
Enter Queue
Doctor, Busy?
Yes?
Wait for a minute
Then ask again
No?
DeQueue
Enter to the Doctor
Doctor

Our Team

Clinic Simulation

أحمد السيد أحمد البهجي

محمد أحمد محمد أحمد عطية

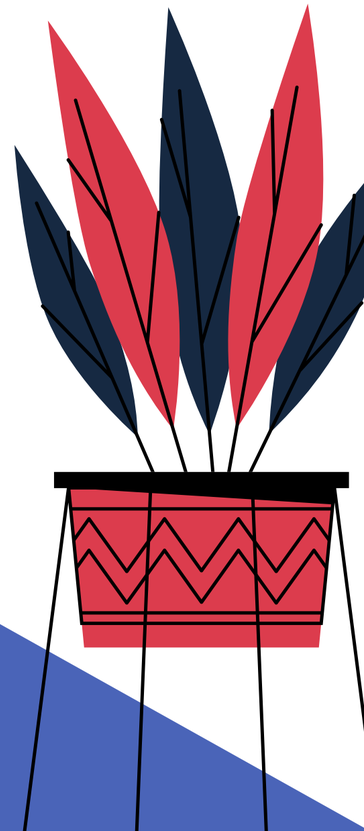
محمد أحمد محمد حسن الجزار

محمد خالد فتحي محمود بدر

محمد خالد محمد محمد إبراهيم العبادي

مهند محمد السيد عبد الكريم

مؤمن أحمد طه الشعراوي



CLINIC SIMULATION BY PYTHON

Questions or comments?

Let’s discuss the simulation! 😊