ROLL NUMBER
CS_063 (Areeba Seher)
CS_061 (Mehar Fatima)
CS_036 (Kinza Hameed)

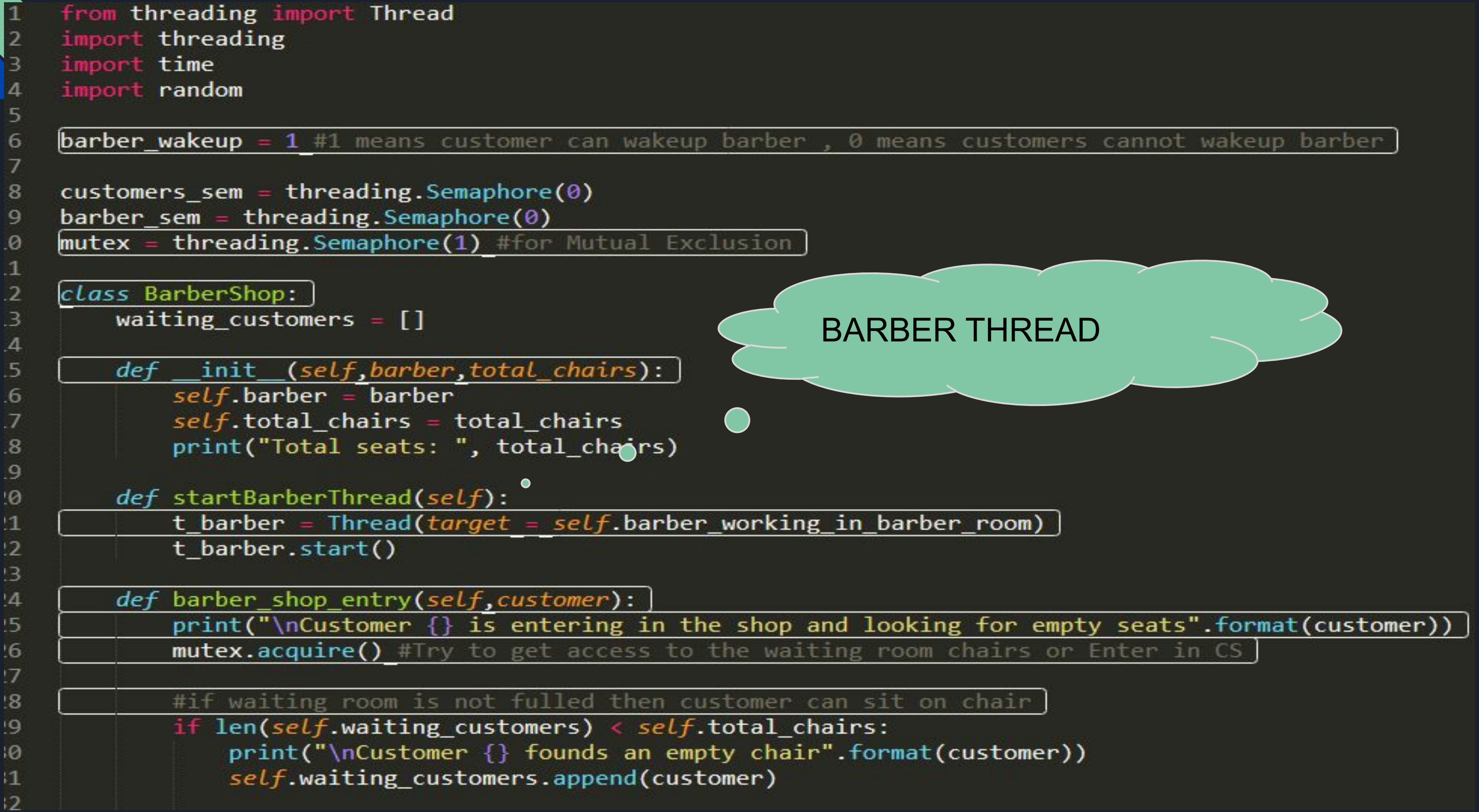
Complex Engineering Activity:

Classical Synchronization Problems
THE BARBERSHOP PROBLEM

CODE SNIPPETS

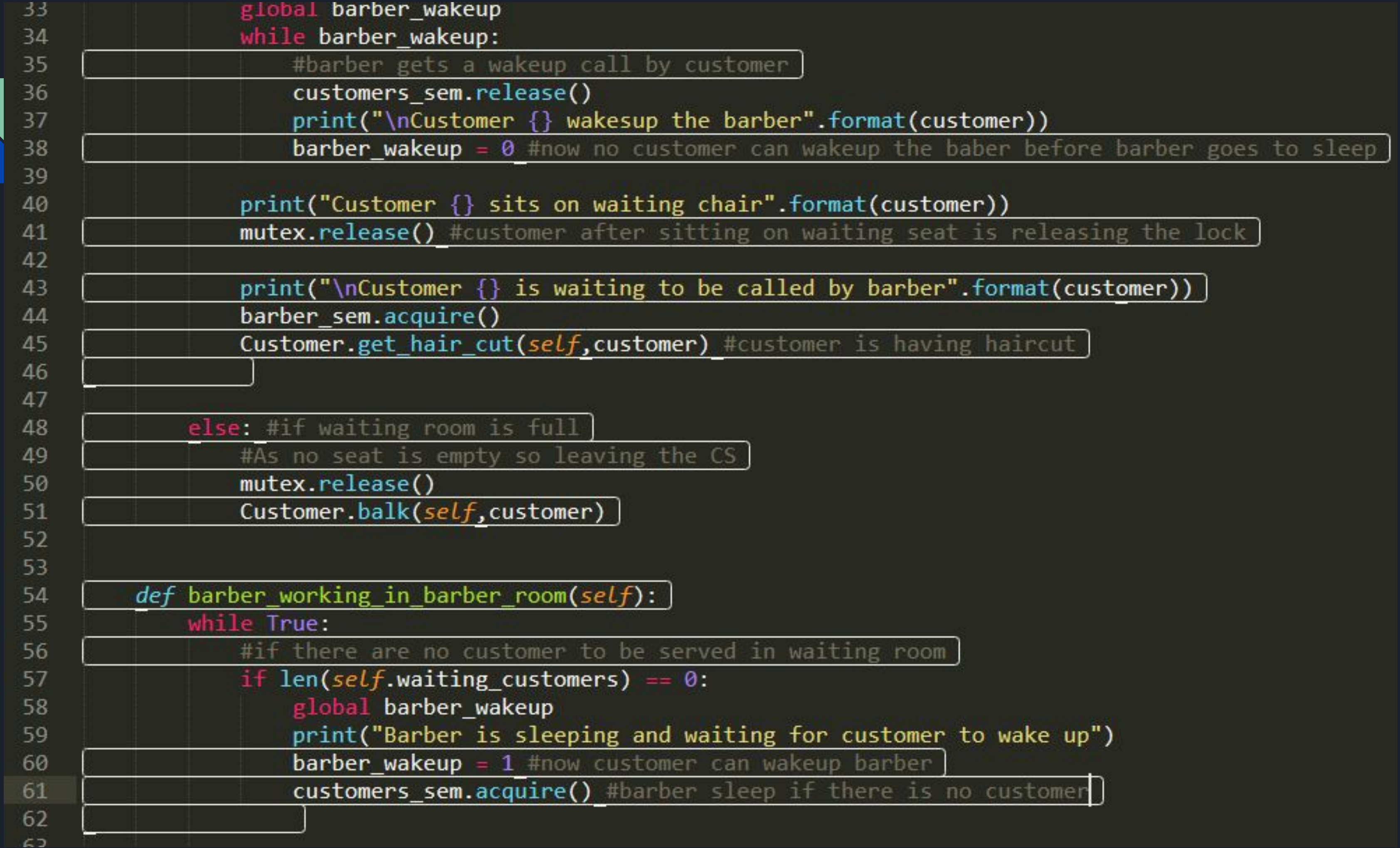
We are implementing this problem in PYTHON

CODE

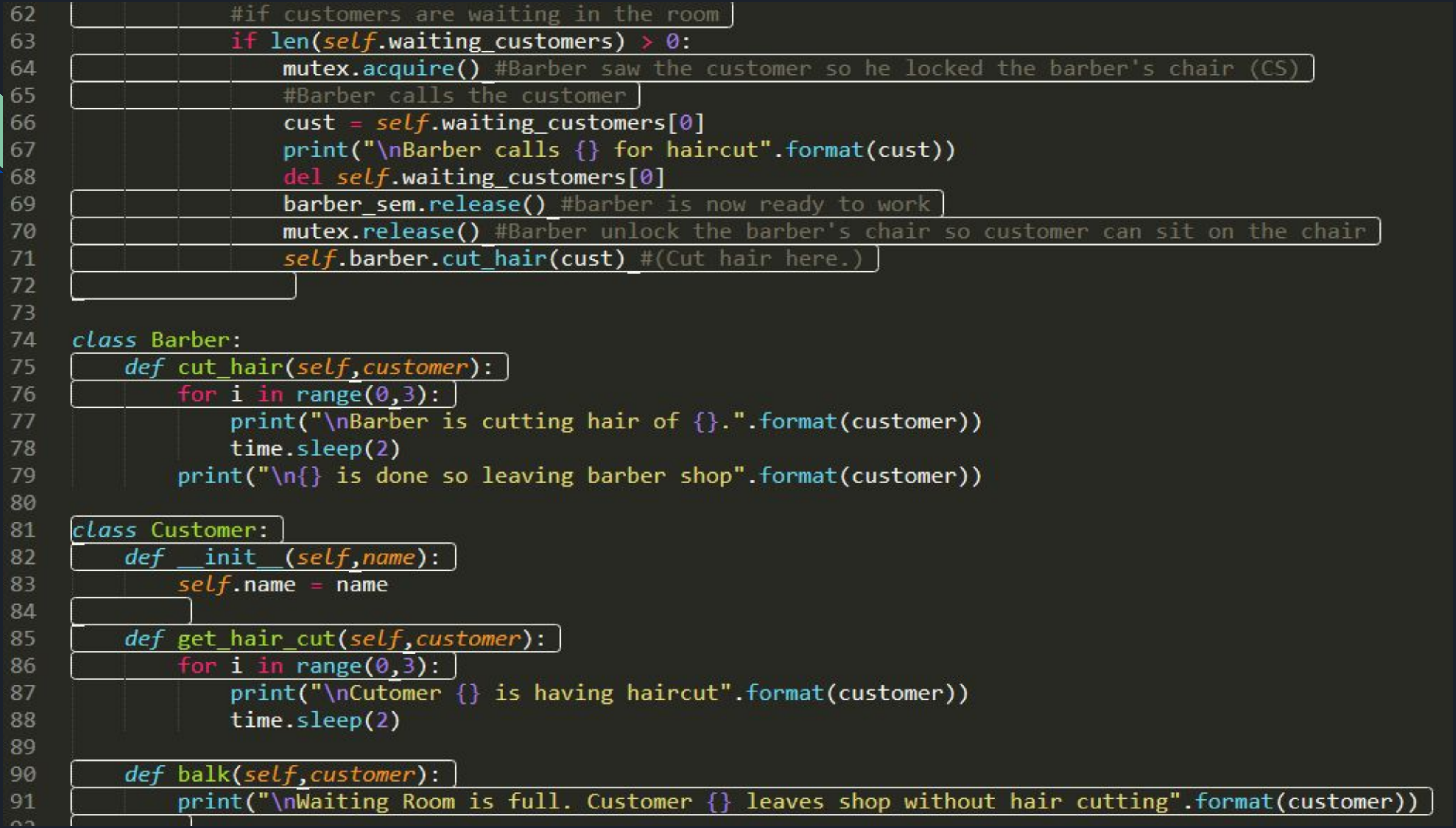


```
from threading import Thread
import threading
import time
import random

barber_wakeup = 1 #1 means customer can wakeup barber , 0 means customers cannot wakeup barber

customers_sem = threading.Semaphore(0)
barber_sem = threading.Semaphore(0)
mutex = threading.Semaphore(1) #for Mutual Exclusion

class BarberShop:
    waiting_customers = []

    def __init__(self,barber,total_chairs):
        self.barber = barber
        self.total_chairs = total_chairs
        print("Total seats: ", total_chairs)

    def startBarberThread(self):
        t_barber = Thread(target = self.barber_working_in_barber_room)
        t_barber.start()

    def barber_shop_entry(self,customer):
        print("\nCustomer {} is entering in the shop and looking for empty seats".format(customer))
        mutex.acquire() #Try to get access to the waiting room chairs or Enter in CS

        #if waiting room is not fulled then customer can sit on chair
        if len(self.waiting_customers) < self.total_chairs:
            print("\nCustomer {} founds an empty chair".format(customer))
            self.waiting_customers.append(customer)

```

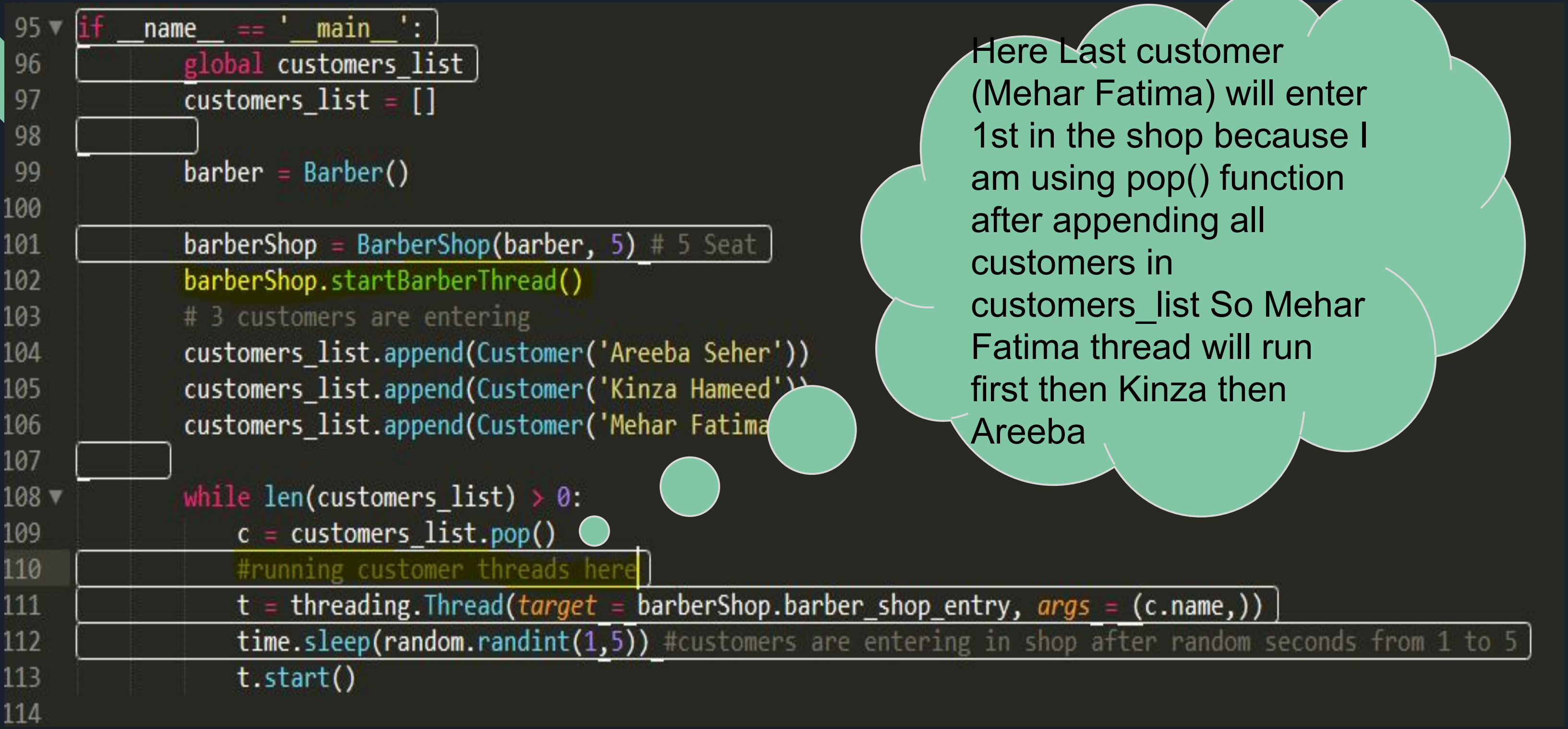
BARBER THREAD



```
            global barber_wakeup
            while barber_wakeup:
                #barber gets a wakeup call by customer
                customers_sem.release()
                print("\nCustomer {} wakesup the barber".format(customer))
                barber_wakeup = 0 #now no customer can wakeup the baber before barber goes to sleep

            print("Customer {} sits on waiting chair".format(customer))
            mutex.release() #customer after sitting on waiting seat is releasing the lock

            print("\nCustomer {} is waiting to be called by barber".format(customer))
            barber_sem.acquire()
            Customer.get_hair_cut(self,customer) #customer is having haircut


        else: #if waiting room is full
            #As no seat is empty so leaving the CS
            mutex.release()
            Customer.balk(self,customer)


    def barber_working_in_barber_room(self):
        while True:
            #if there are no customer to be served in waiting room
            if len(self.waiting_customers) == 0:
                global barber_wakeup
                print("Barber is sleeping and waiting for customer to wake up")
                barber_wakeup = 1 #now customer can wakeup barber
                customers_sem.acquire() #barber sleep if there is no customer

```



```
            #if customers are waiting in the room
            if len(self.waiting_customers) > 0:
                mutex.acquire() #Barber saw the customer so he locked the barber's chair (CS)
                #Barber calls the customer
                cust = self.waiting_customers[0]
                print("\nBarber calls {} for haircut".format(cust))
                del self.waiting_customers[0]
                barber_sem.release() #barber is now ready to work
                mutex.release() #Barber unlock the barber's chair so customer can sit on the chair
                self.barber.cut_hair(cust) #(Cut hair here.)


class Barber:
    def cut_hair(self,customer):
        for i in range(0,3):
            print("\nBarber is cutting hair of {}.".format(customer))
            time.sleep(2)
        print("\n{} is done so leaving barber shop".format(customer))

class Customer:
    def __init__(self,name):
        self.name = name

    def get_hair_cut(self,customer):
        for i in range(0,3):
            print("\nCutomer {} is having haircut".format(customer))
            time.sleep(2)

    def balk(self,customer):
        print("\nWaiting Room is full. Customer {} leaves shop without hair cutting".format(customer))
```



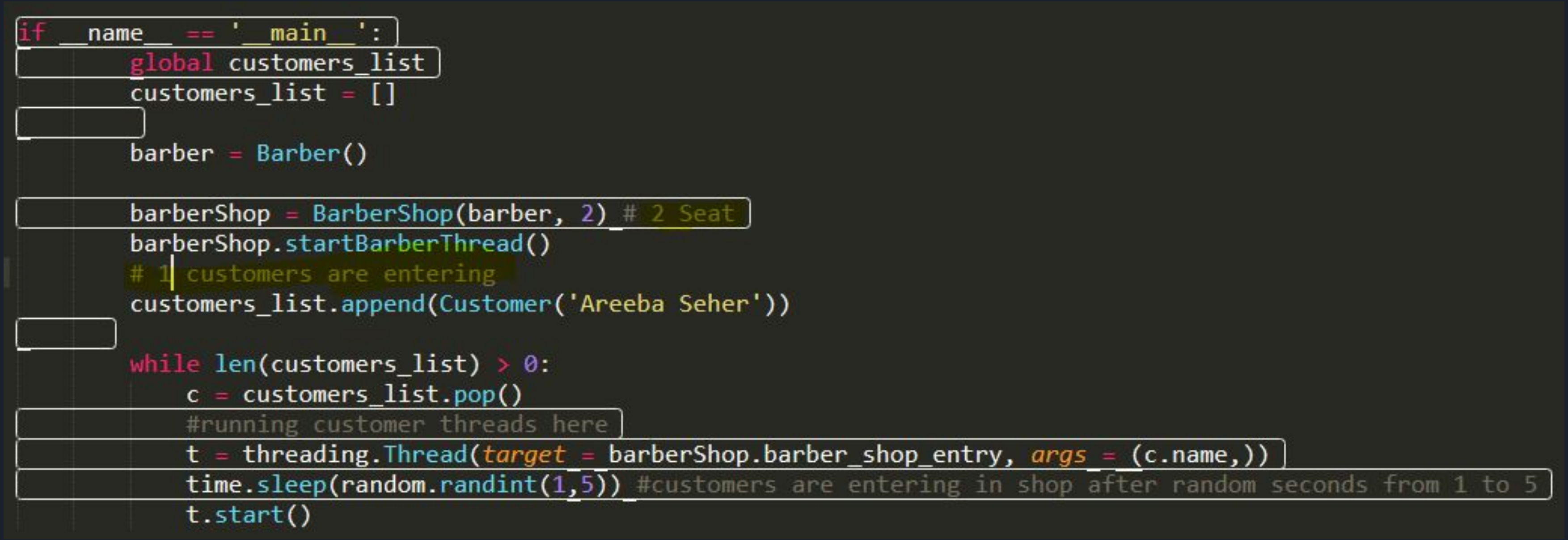
```python
if __name__ == '__main__':
    global customers_list
    customers_list = []

    barber = Barber()

    barberShop = BarberShop(barber, 5) # 5 Seat
    barberShop.startBarberThread()
    # 3 customers are entering
    customers_list.append(Customer('Areeba Seher'))
    customers_list.append(Customer('Kinza Hameed'))
    customers_list.append(Customer('Mehar Fatima

    while len(customers_list) > 0:
        c = customers_list.pop()
        #running customer threads here
        t = threading.Thread(target = barberShop.barber_shop_entry, args = (c.name,))
        time.sleep(random.randint(1,5)) #customers are entering in shop after random seconds from 1 to 5
        t.start()
```

Here Last customer (Mehar Fatima) will enter 1st in the shop because I am using pop() function after appending all customers in customers_list So Mehar Fatima thread will run first then Kinza then Areeba

TEST CASE # 1 (INPUTS)

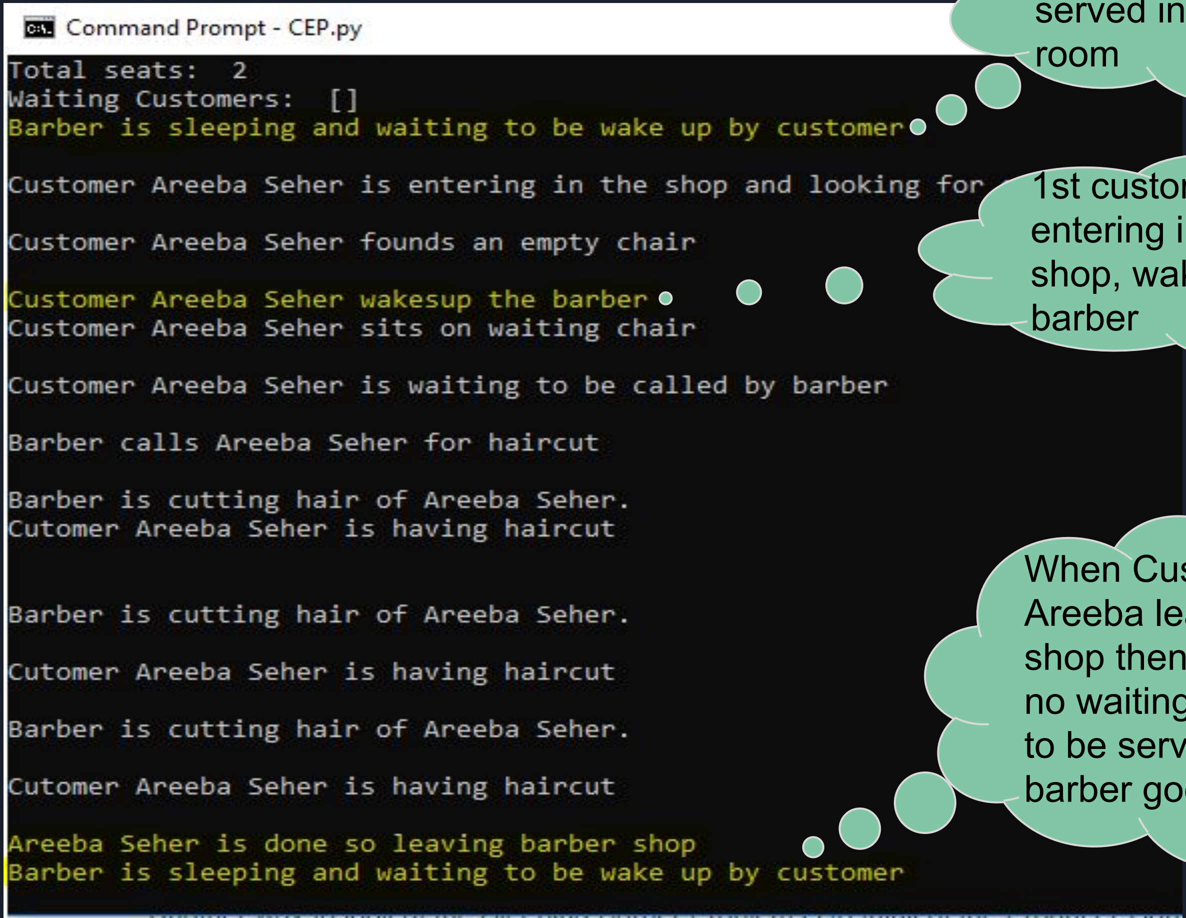
If there are no customers to be served, the barber goes to sleep AND If the barber is asleep, the customer wakes up the barber.

In this case Total seats are 2 and only 1 customer is entering in the barber shop and waiting customer list is empty.



```
if __name__ == '__main__':
        global customers_list
        customers_list = []

        barber = Barber()

        barberShop = BarberShop(barber, 2) # 2 Seat
        barberShop.startBarberThread()
        # 1 customers are entering
        customers_list.append(Customer('Areeba Seher'))

        while len(customers_list) > 0:
            c = customers_list.pop()
            #running customer threads here
            t = threading.Thread(target = barberShop.barber_shop_entry, args = (c.name,))
            time.sleep(random.randint(1,5)) #customers are entering in shop after random seconds from 1 to 5
            t.start()
```

TEST CASE # 1 OUTPUT


Command Prompt - CEP.py

```
Total seats:  2
Waiting Customers:  []
Barber is sleeping and waiting to be wake up by customer

Customer Areeba Seher is entering in the shop and looking for

Customer Areeba Seher founds an empty chair

Customer Areeba Seher wakesup the barber
Customer Areeba Seher sits on waiting chair

Customer Areeba Seher is waiting to be called by barber

Barber calls Areeba Seher for haircut

Barber is cutting hair of Areeba Seher.
Cutomer Areeba Seher is having haircut


Barber is cutting hair of Areeba Seher.

Cutomer Areeba Seher is having haircut

Barber is cutting hair of Areeba Seher.

Cutomer Areeba Seher is having haircut

Areeba Seher is done so leaving barber shop
Barber is sleeping and waiting to be wake up by customer
```

Barber is sleeping as there are no customer to be served in waiting room

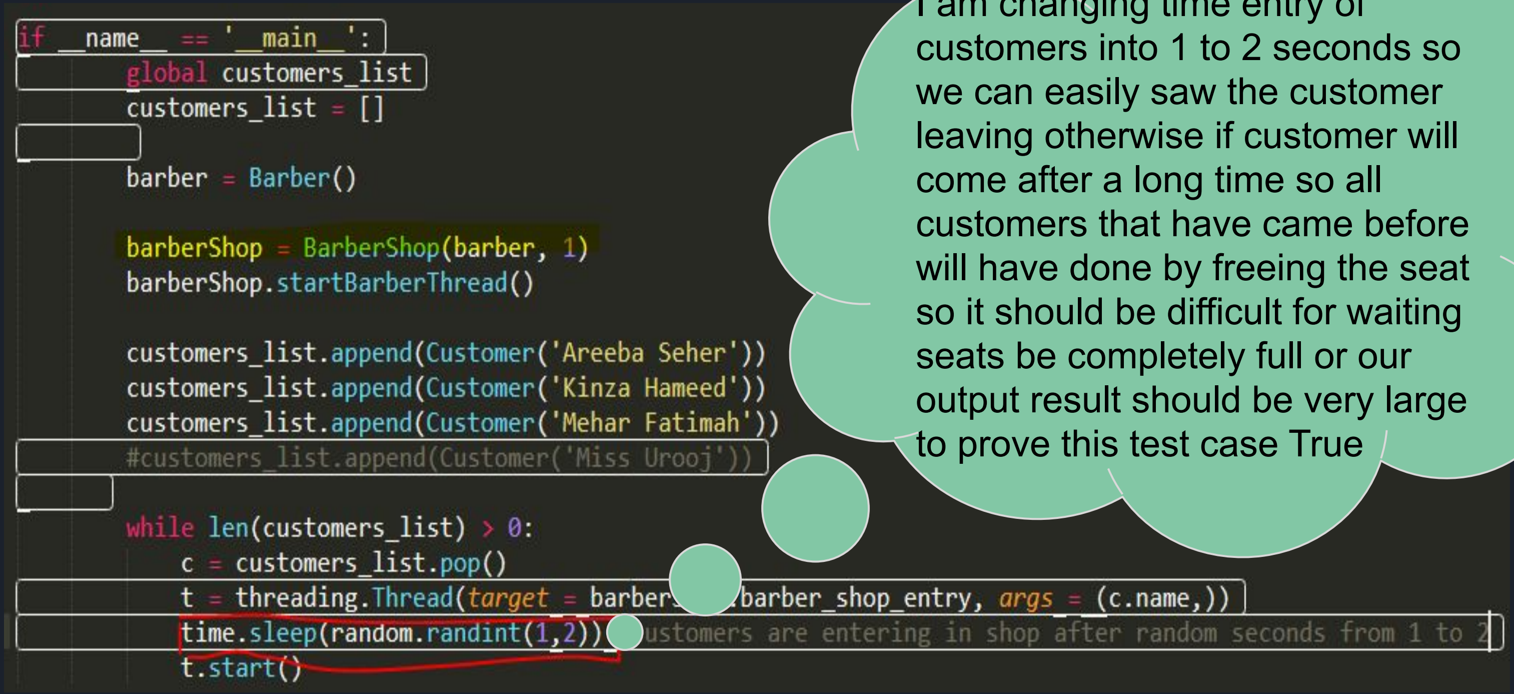
1st customer after entering in the barber shop, wake up the barber

When Customer Areeba leaves the shop then there are no waiting customer to be served so barber goes to sleep

TEST CASE # 2 (INPUTS)

If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop.

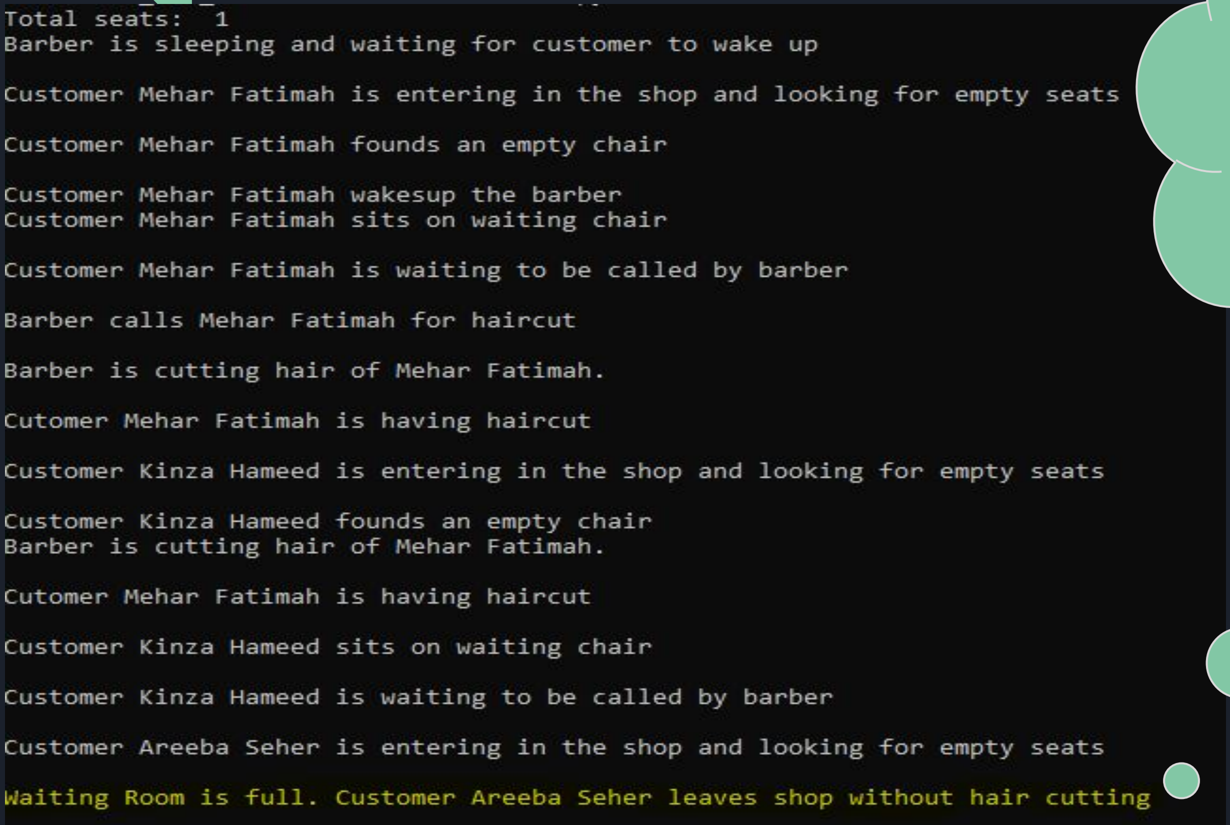
Here Total seats are one and 3 customers are entering in shop



```
if __name__ == '__main__':
        global customers_list
        customers_list = []

        barber = Barber()

        barberShop = BarberShop(barber, 1)
        barberShop.startBarberThread()

        customers_list.append(Customer('Areeba Seher'))
        customers_list.append(Customer('Kinza Hameed'))
        customers_list.append(Customer('Mehar Fatimah'))
        #customers_list.append(Customer('Miss Urooj'))

        while len(customers_list) > 0:
            c = customers_list.pop()
            t = threading.Thread(target = barber   barber_shop_entry, args = (c.name,))
            time.sleep(random.randint(1,2))   ustomers are entering in shop after random seconds from 1 to 2
            t.start()
```

I am changing time entry of customers into 1 to 2 seconds so we can easily saw the customer leaving otherwise if customer will come after a long time so all customers that have came before will have done by freeing the seat so it should be difficult for waiting seats be completely full or our output result should be very large to prove this test case True

TEST CASE # 2 (OUTPUT)



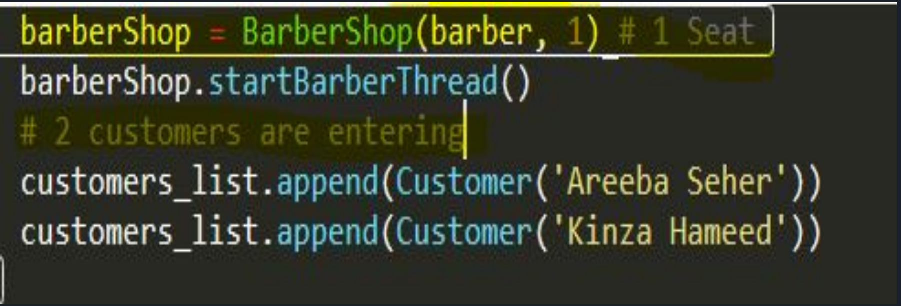
```
Total seats:  1
Barber is sleeping and waiting for customer to wake up

Customer Mehar Fatimah is entering in the shop and looking for empty seats

Customer Mehar Fatimah founds an empty chair

Customer Mehar Fatimah wakesup the barber
Customer Mehar Fatimah sits on waiting chair

Customer Mehar Fatimah is waiting to be called by barber

Barber calls Mehar Fatimah for haircut

Barber is cutting hair of Mehar Fatimah.

Cutomer Mehar Fatimah is having haircut

Customer Kinza Hameed is entering in the shop and looking for empty seats

Customer Kinza Hameed founds an empty chair
Barber is cutting hair of Mehar Fatimah.

Cutomer Mehar Fatimah is having haircut

Customer Kinza Hameed sits on waiting chair

Customer Kinza Hameed is waiting to be called by barber

Customer Areeba Seher is entering in the shop and looking for empty seats

Waiting Room is full. Customer Areeba Seher leaves shop without hair cutting
```

Waiting seats = 0

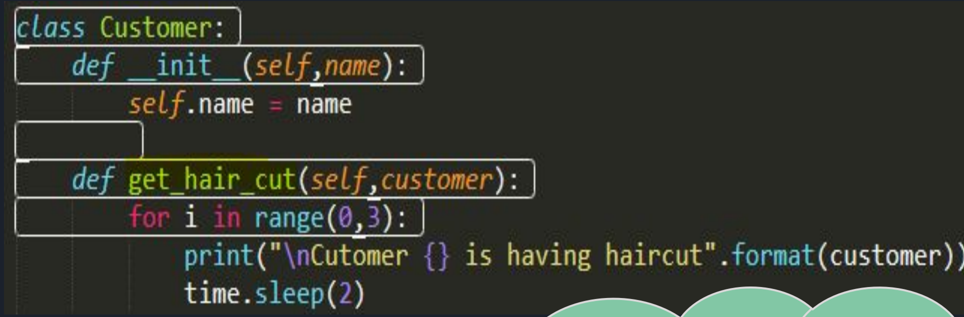
- Meher comes , sit on w_seat, w_seats = 1
- Meher is in barber room (w_seat = 0)
- Kinza comes, sit on w_seat w_seats = 1
- Areeba Seher comes , w_seat is already 1 so leaving the shop by invoking balk function

TEST CASE # 3

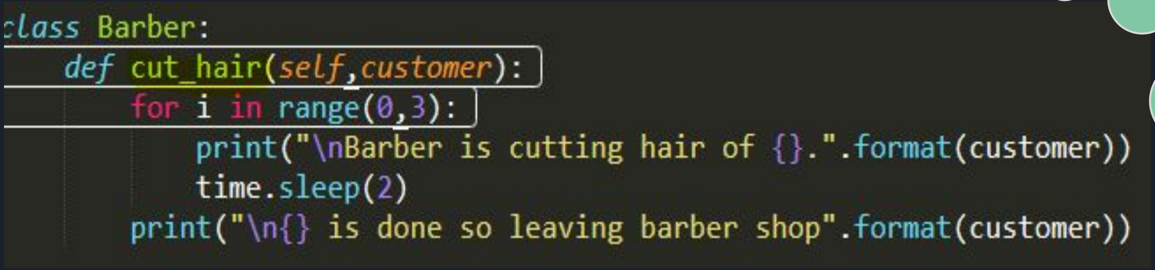
When the barber invokes cutHair, there should be exactly one thread invoking getHairCut concurrently. AND If the barber is busy, but chairs are available, then the customer sits in one of the free chairs.



```
barberShop = BarberShop(barber, 1) # 1 Seat
barberShop.startBarberThread()
# 2 customers are entering
customers_list.append(Customer('Areeba Seher'))
customers_list.append(Customer('Kinza Hameed'))
```



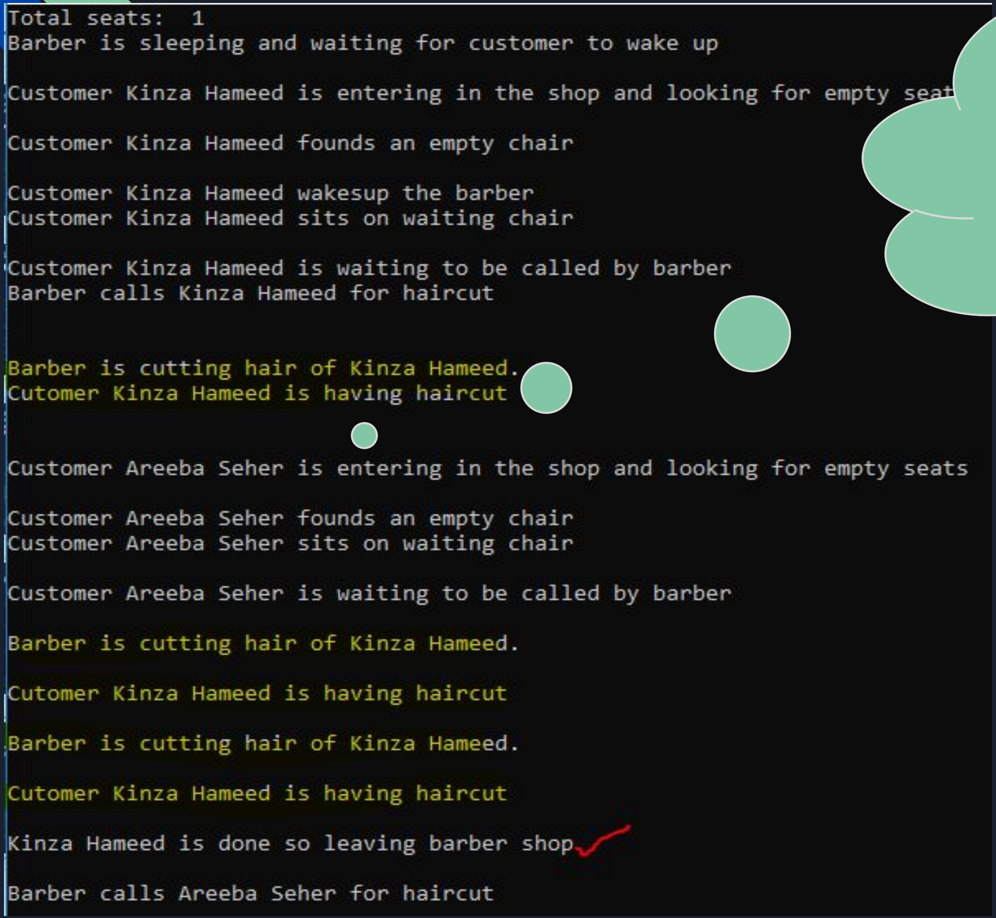
```
class Customer:
    def __init__(self,name):
        self.name = name

    def get_hair_cut(self,customer):
        for i in range(0,3):
            print("\nCutomer {} is having haircut".format(customer)
            time.sleep(2)
```



```
class Barber:
    def cut_hair(self,customer):
        for i in range(0,3):
            print("\nBarber is cutting hair of {}.".format(customer))
            time.sleep(2)
        print("\n{} is done so leaving barber shop".format(customer))
```

I am implementing loop in cutHair and getHairCut to check only that whenever baber is invoking cutHair then getHairCut is running concurrently or not

TEST CASE # 3 (OUTPUT)

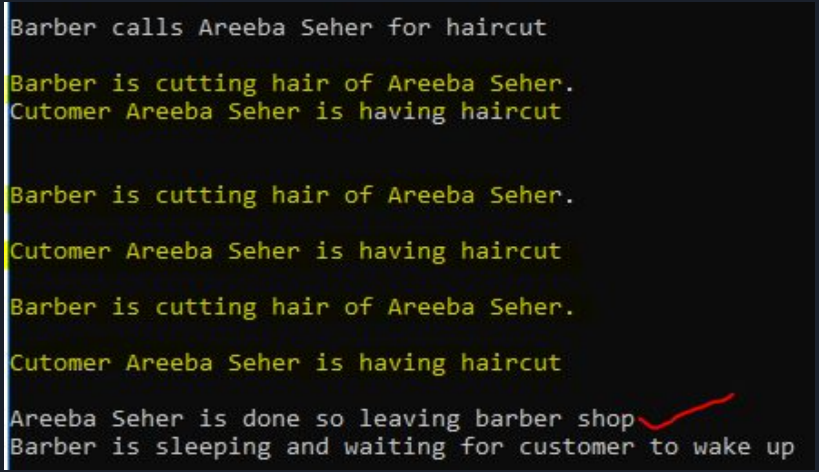


```
Total seats:  1
Barber is sleeping and waiting for customer to wake up

Customer Kinza Hameed is entering in the shop and looking for empty seat

Customer Kinza Hameed founds an empty chair

Customer Kinza Hameed wakesup the barber
Customer Kinza Hameed sits on waiting chair

Customer Kinza Hameed is waiting to be called by barber
Barber calls Kinza Hameed for haircut


Barber is cutting hair of Kinza Hameed.
Cutomer Kinza Hameed is having haircut


Customer Areeba Seher is entering in the shop and looking for empty seats

Customer Areeba Seher founds an empty chair
Customer Areeba Seher sits on waiting chair

Customer Areeba Seher is waiting to be called by barber

Barber is cutting hair of Kinza Hameed.

Cutomer Kinza Hameed is having haircut

Barber is cutting hair of Kinza Hameed.

Cutomer Kinza Hameed is having haircut

Kinza Hameed is done so leaving barber shop

Barber calls Areeba Seher for haircut
```

Barber is busy with Kinza Hameed and in between Areeba Seher comes and found an empty chair in waiting room so she sits on the chair

AND

cutHair and getHairCut are running concurrently.



```
Barber calls Areeba Seher for haircut

Barber is cutting hair of Areeba Seher.
Cutomer Areeba Seher is having haircut


Barber is cutting hair of Areeba Seher.

Cutomer Areeba Seher is having haircut

Barber is cutting hair of Areeba Seher.

Cutomer Areeba Seher is having haircut

Areeba Seher is done so leaving barber shop
Barber is sleeping and waiting for customer to wake up
```

THANK YOU