from threading import Thread

from queue import Queue

import time

def sender\_thread(msg\_queue,messages):

  for m in messages:

    print(f"Sender = {m} is sending")

    msg\_queue.put(m)

    time.sleep(2)

  msg\_queue.put(None)

def receiver\_thread(msg\_queue):

  while True:

    message = msg\_queue.get()

    if message is None:

      break

    print(f"Receiver = {message} is receiving")

def main():

  queue\_obj = Queue()

  messages = ["Hello","How Are You?","Whats Up"]

  sender = Thread(target=sender\_thread,args=(queue\_obj,messages))

  receiver = Thread(target=receiver\_thread,args=(queue\_obj,))

  sender.start()

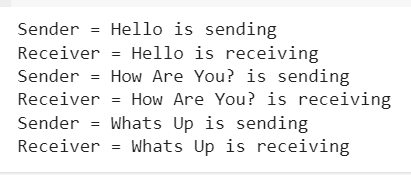
  receiver.start()

  sender.join()

  receiver.join()

if \_\_name\_\_ == "\_\_main\_\_":

    main()



from threading import Thread,Lock

from queue import Queue

import time

import random

class Restaurant:

  def \_\_init\_\_(self):

    self.orders\_to\_prepare = Queue()

    self.kitchen\_lock = Lock()

    self.kitchen\_prepared\_orders = []

  def place\_order(self,order):

    self.orders\_to\_prepare.put(order)

    print(f"{order} is placed")

  def prepare\_order(self):

    while True:

      time.sleep(random.uniform(1,4))

      with self.kitchen\_lock:

        if not self.orders\_to\_prepare.empty():

          result = self.orders\_to\_prepare.get()

          print(f"Chef is preparing {result}")

          self.kitchen\_prepared\_orders.append(result)

        else:

          break;

  def serve\_order(self):

    while True:

      time.sleep(random.uniform(1,3))

      with self.kitchen\_lock:

        if self.kitchen\_prepared\_orders:

          result = self.kitchen\_prepared\_orders.pop(0)

          print(f"Waiter is serving {result}")

        else:

          if not self.orders\_to\_prepare:

            break;

def main():

  rest\_obj = Restaurant()

  chef\_threads = [Thread(target=rest\_obj.prepare\_order) for \_ in range(3)]

  waiter\_threads = [Thread(target=rest\_obj.serve\_order) for \_ in range(3)]

  for chef\_thread in chef\_threads:

    chef\_thread.start()

  for waiter\_thread in waiter\_threads:

    waiter\_thread.start()

  for i in range(5):

    rest\_obj.place\_order(f"Order : {i+1}")

  for chef\_thread in chef\_threads:

    chef\_thread.join()

  for waiter\_thread in waiter\_threads:

    waiter\_thread.join()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

A screenshot of a menu

Description automatically generated