**Домашнее задание по теме "Очереди для обмена данными между потоками."**

**Цель задания:**

Освоить механизмы создания потоков и очередей для обмена данных между ними в Python.  
Практически применить знания, создав и запустив несколько потоков и очередь.

**Задание:**

Моделирование работы сети кафе с несколькими столиками и потоком посетителей, прибывающих для заказа пищи и уходящих после завершения приема.  
  
Есть сеть кафе с несколькими столиками. Посетители приходят, заказывают еду, занимают столик, употребляют еду и уходят. Если столик свободен, новый посетитель принимается к обслуживанию, иначе он становится в очередь на ожидание.  
  
Создайте 3 класса:  
Table - класс для столов, который будет содержать следующие атрибуты: number(int) - номер стола, is\_busy(bool) - занят стол или нет.  
  
Cafe - класс для симуляции процессов в кафе. Должен содержать следующие атрибуты и методы:

1. Атрибуты queue - очередь посетителей (создаётся внутри init), tables список столов (поступает из вне).
2. Метод customer\_arrival(self) - моделирует приход посетителя(каждую секунду).
3. Метод serve\_customer(self, customer) - моделирует обслуживание посетителя. Проверяет наличие свободных столов, в случае наличия стола - начинает обслуживание посетителя (запуск потока), в противном случае - посетитель поступает в очередь. Время обслуживания 5 секунд.

Customer - класс (поток) посетителя. Запускается, если есть свободные столы.  
  
Так же должны выводиться текстовые сообщения соответствующие событиям:

1. Посетитель номер <номер посетителя> прибыл.
2. Посетитель номер <номер посетителя> сел за стол <номер стола>. (начало обслуживания)
3. Посетитель номер <номер посетителя> покушал и ушёл. (конец обслуживания)
4. Посетитель номер <номер посетителя> ожидает свободный стол. (помещение в очередь)

**Пример работы:**

# Создаем столики в кафе  
table1 = Table(1)  
table2 = Table(2)  
table3 = Table(3)  
tables = [table1, table2, table3]  
  
# Инициализируем кафе  
cafe = Cafe(tables)  
  
# Запускаем поток для прибытия посетителей  
customer\_arrival\_thread = threading.Thread(target=cafe.customer\_arrival)  
customer\_arrival\_thread.start()  
  
# Ожидаем завершения работы прибытия посетителей  
customer\_arrival\_thread.join()

**Вывод на консоль (20 посетителей [ограничение выставить в методе customer\_arrival]):**

Посетитель номер 1 прибыл  
Посетитель номер 1 сел за стол 1  
Посетитель номер 2 прибыл  
Посетитель номер 2 сел за стол 2  
Посетитель номер 3 прибыл  
Посетитель номер 3 сел за стол 3  
Посетитель номер 4 прибыл  
Посетитель номер 4 ожидает свободный стол  
Посетитель номер 5 прибыл  
Посетитель номер 5 ожидает свободный стол  
**......**  
Посетитель номер 20 прибыл  
Посетитель номер 20 ожидает свободный стол  
Посетитель номер 17 покушал и ушёл.  
Посетитель номер 20 сел за стол N.  
Посетитель номер 18 покушал и ушёл.  
Посетитель номер 19 покушал и ушёл.  
Посетитель номер 20 покушал и ушёл.

**Файл с кодом прикрепите к домашнему заданию.**

**Решение**

import threading # модуль для работы с многопоточностью.  
import time # модуль для работы со временем.  
  
import queue # модуль для работы с очередями.  
  
# Создаем 3 класса:  
class Table: # Класс Table содержит атрибуты number (номер стола) и is\_busy (занят стол или нет).  
 def \_\_init\_\_(self, number):  
 self.number = number  
 self.is\_busy = False  
  
class Customer(threading.Thread): # Класс Customer наследуется от threading.Thread/ класс (поток) посетителя.  
 # Запускается, если есть свободные столы.  
 # и содержит атрибуты:  
 def \_\_init\_\_(self, number, table):  
 super().\_\_init\_\_()  
 self.number = number # number (номер посетителя)  
 self.table = table # table (стол, за которым сидит посетитель)  
  
 def run(self): # Метод run запускается при запуске потока и моделирует приход посетителя,  
 # прием пищи и уход.  
  
 print(f"Посетитель номер {self.number} сел за стол {self.table.number}.") #(начало обслуживания)  
 time.sleep(5)  
 print(f"Посетитель номер {self.number} покушал и ушёл.") # (начало обслуживания)  
  
class Cafe: # Класс Cafe- класс для симуляции процессов в кафе/содержит атрибуты:  
  
 def \_\_init\_\_(self, tables):  
 self.tables = tables # tables (список столов)/ Атрибут tables список столов (поступает из вне)  
 self.queue = queue.Queue() # queue (очередь посетителей) Атрибут queue - очередь посетителей (создаётся внутри init)  
 self.customer\_number = 1 # customer\_number (номер текущего посетителя)  
  
 def customer\_arrival(self): # Метод customer\_arrival(self) - Этот метод моделирует прибытие посетителей в кафе,  
 # назначение им свободных столов или помещение их в очередь в случае, если все столики заняты.  
 while self.customer\_number <= 20: # Цикл продолжается, пока количество посетителей не достигнет 20 (ограничение на количество посетителей).  
 time.sleep(1) # Программа ждет 1 секунду перед обработкой следующего посетителя.  
 # Это моделирует приход посетителей с интервалом в 1 секунду.  
 table = next((t for t in self.tables if not t.is\_busy), None) # Ищет первый свободный столик из списка self.tables.  
 # Если все столики заняты, возвращает None.  
 if table: # Проверяет, найден ли свободный столик.  
 table.is\_busy = True # Устанавливает флаг is\_busy для выбранного столика как занятый.  
 customer = Customer(self.customer\_number, table) # Создает экземпляр класса Customer для текущего посетителя  
 # с номером self.customer\_number и выбранным столиком.  
 customer.start() # Запускает поток для нового посетителя, который выполняет метод run класса Customer.  
 self.customer\_number += 1 # Увеличивает номер следующего посетителя для контроля количества посетителей.  
 else: # Если не удалось найти свободный столик, выводит сообщение о том, что посетитель ожидает свободный стол.  
 print(f"Посетитель номер {self.customer\_number} ожидает свободный стол.") # помещение в очередь  
 self.customer\_number += 1  
  
# Создаем столики в кафе  
table1 = Table(1)  
table2 = Table(2)  
table3 = Table(3)  
tables = [table1, table2, table3]  
  
# Инициализируем кафе  
cafe = Cafe(tables)  
  
# Запускаем поток для прибытия посетителей  
customer\_arrival\_thread = threading.Thread(target=cafe.customer\_arrival)  
customer\_arrival\_thread.start()  
  
# Ожидаем завершения работы прибытия посетителей  
customer\_arrival\_thread.join()

ВЫВОД:

Посетитель номер 1 сел за стол 1.

Посетитель номер 2 сел за стол 2.

Посетитель номер 3 сел за стол 3.

Посетитель номер 4 ожидает свободный стол.

Посетитель номер 5 ожидает свободный стол.

Посетитель номер 1 покушал и ушёл.

Посетитель номер 6 ожидает свободный стол.

Посетитель номер 2 покушал и ушёл.

Посетитель номер 7 ожидает свободный стол.

Посетитель номер 3 покушал и ушёл.

Посетитель номер 8 ожидает свободный стол.

Посетитель номер 9 ожидает свободный стол.

Посетитель номер 10 ожидает свободный стол.

Посетитель номер 11 ожидает свободный стол.

Посетитель номер 12 ожидает свободный стол.

Посетитель номер 13 ожидает свободный стол.

Посетитель номер 14 ожидает свободный стол.

Посетитель номер 15 ожидает свободный стол.

Посетитель номер 16 ожидает свободный стол.

Посетитель номер 17 ожидает свободный стол.

Посетитель номер 18 ожидает свободный стол.

Посетитель номер 19 ожидает свободный стол.

Посетитель номер 20 ожидает свободный стол.

Process finished with exit code 0