159.172 Assignment 2

Worth 20% Due Sunday 8pm, Oct.9th 2017

Part 1 – The Hospital Simulation – 14 marks

The problem for this assignment is to build queuing model of a small hospital with several doctors, and patients arriving at various times throughout the day, to get an idea of how long they have to wait, and the maximum number of people in the waiting room at any time.

Making people for Doctors and Patients

You're going to need some people, both doctors and patients. I've provided a class called *Person()* (in *person.py* on Stream) that will create random people for you, complete with names, ages and occupations. You can specify many parameters when creating a person

```
e.g. p1 = person.Person(first_name = 'Fred', age = 27, occupation='Cook')
     p2 = person.Person()
```

Any parameters that aren't specified are chosen randomly. None are required. Save the *person.py* to the same directory as your main file, and then import the person module (*import person)*.

Patients are seen in order of their arrival (it's a queue) and randomly allocated to one of the free doctors (if there's more than one free), otherwise they have to wait.

Handling Time in the Simulation

This program is a little different to most of the programs you've written. It doesn't need any user input but does have an explicit concept of time. You set up the initial conditions, and then just let it run, similar to the *Game of Life*. At each time-step, patients might arrive or they might not. If a doctor's free, they'll get to see the doctor immediately, otherwise they're put in a queue. When a doctor finishes with their current patient, the first person in the queue gets to see that doctor. If more than one doctor is free, the doctor to handle a patient is chosen randomly.

Time is handled in the program as a simple integer, starting from 0 and incrementing to 660, which is the number of minutes from 1pm until midnight. Start running at 1pm.

When displaying time, display both the absolute timestep (ranging from 0 to 660), and the time after 1pm, so *t = 15* is 1:15pm, *t = 67* is 2.07pm etc.

Here, we can use an zero-padding type of formatting, to ensure that numbers from 0-9 get displayed as 00, 01, 02 … 09, so the '3:04' is displayed not '3:4' or '3: 4'. Note the extra zero (0) in the minutes position:

```
start_time = 1   # so t = 0 is 1pm, t is the time-step
s = '(%-3d) %2d:%02dpm ' % (t, hours + start_time, minutes)
```

What's wrong the patients?

The patients have various ailments (things that are wrong with them), and these are specified with an average waiting time in minutes:

```
ailments = [('cold', 10), ('broken leg', 45), ('measles', 15),
            ('pneumonia', 30), ('cough', 10), ('asthma',  12),
            ('food poisoning', 18), ('insect bite', 9),
            ('meningitis', 30), ('stroke', 25)]
```

Feel free at add to this list to make it interesting.

Making the length of each appointment vary

The actual time of each person's appointment will vary, both because of what's wrong with them and how long the diagnosis takes. I've modelled this with this function that uses a Gaussian distribution function from the *random* module.

The function *random.gauss(mean, stdDev)* returns floating point number that varies around the specified mean with a normal distribution. I've arbitrarily decided that if the mean (average) appointment time is 10 minutes, then it should shouldn't be less that half this (5) nor longer than half as much again (15). The following function does this.

```
def estimate_appointment_length(mean_time):
    time = mean_time
    return int(time/2 + random.gauss(time/2, time/4))
```

For example, with *mean_time* as 10, this function returns numbers centered around 10, but that can range from 5 to 15, which is a little more realistic, as appointment times do vary.

Patients

Patients are an instance of *Person().* At each timestep, patients arrive with a small probability: 5-10% seems about right, and more than one, say 1 to 3 may arrive at the same time.

Each patient has a randomly chosen ailment, and an estimated appointment length based on the average time for that ailment.

You'll need to record when the patient arrives, so you can calculate how long they've been waiting.

15 Minute summary statistics

Two other statistics are necessary, the maximum number of patients that have been in the queue any time during the day, and the maximum length of time a patient had to wait at any time during the day. These are displayed every 15 minutes, as you can see on the sample simulation run included later.

Doctors

There are a small number of doctors (1 – 5 probably) and is fixed for each simulation run. Doctors are an instance of a Person(), so they have a name, age and occupation. In the example shown later, only the name is used.

It's useful but not required for each doctor to have an *is_busy* attribute which is *true* if they're currently allocated to a patient, and *false* otherwise. If they're allocated a patient, you'll need to know *when the appointment should finish* and you'll also need to remember *the patient* (which is an instance of *person) s*o you can show their name when the appointment finishes. This end time set from the time step when the patient is initially allocated to a doctor, and the length of the estimated appointment (calculated for each patient).

Treating time as a simple integer makes two things quite easy: setting the end time (it's just addition) and determining whether, at each timestep, the appointment is over.

The general flow of the program is something like:

```
# set up the doctors

for each timestep:
    # have any patients arrived
    # are any doctors free?
    # can a patient see a doctor?
    # is it time to print some statistics
```

Class Structures

The class **Person()** has been supplied. Make use of this when creating a class for **Doctor** and a class for **Patient**, each of which stores whatever data and methods you think appropriate.

The Queue Class

You MUST create a *Queue* Class in separate module (file) and then import it. You can use any data structure you like to implement the queue.

In addition to the usual queue methods:

.**add**(*item*) or **.enqueue**(*item*)	add an item to the end of the queue
.dequeue()	remove & return the item from front of the queue
.isEmpty()	returns *True* if the queue is empty …
.queue_length()	returns the number of items in the queue

Add

.display()	so you can display names of the patients waiting

Part 2: Enhancing the simulation – 6 marks

The current simulation is very simple. It assumes that:

- **the likelihood of people arriving is uniform throughout the day,** but things are usually quieter in the early morning and at night.
- **doctors don't need breaks: but this would be dangerous, as doctors get tired too.** Fix this by temporarily remove doctor(s) from service every few hours.
- **all patients have an same priority.** This isn't correct. Someone with symptoms of a heart-attack won't have to wait behind someone with a cold.

  This could be handled by having multiple queues (e.g. urgent/ordinary) or using a priority queue, but we haven't covered priority queues yet.
- **a patient's ailments are the same whatever the age of the person**. In practise this isn't so. Children at school and the elderly (70+) get sick more often.
- **Better Statistics:** the maximum number of people waiting and the maximum waiting time will vary from one simulation to the next, as the arrival of patients varies probabilistically, as to the number of patients arriving at any time, and the length of their appointments. You could run the simulation multiple times to calculate the mean (and standard deviation?) of the above statistics.

For the 3 marks, choose **ONE** of these and incorporate it into your simulation. **SPECIFY WHICH ONE**. You're welcome to add more than one, but if so, specify the "MAIN" one.

Changing the scenario

I've framed the simulation around a small hospital. If you prefer, you can modify the scenario to something else as long as there are:

- multiple servers (like the doctors)
- randomly arriving events that have to be handled (like the patients)
- more than one event (patient) might arrive at each timestep.
- the time to handle each event varies probabilistically, like *estimate_appointment_time()*
- the same summary statistics produced every 15 minutes

What to submit on Stream

The supplied file *person.py* shouldn't need changing, so upload the two files:

1. your **queues.py** module
2. the **.py** file containing your code for this simulation.
3. **BOTH FILES MUST CONTAIN/DISPLAY YOUR NAME AND MASSEY ID**

Sample Simulation Output – your output DOESN'T have to look exactly the same

```
======= Doctors on Duty ========
Dr Skyla Gomez
Dr Eli Ware
Dr Faith Jennings
```

… some timesteps deleted …

```
================================================================================
(60 )  2:00pm  3 doctors free, 0 patients waiting, Longest queue: 2,
                Longest wait: 1 minutes
================================================================================
(69 )  2:09pm  Arrival: Dr Nathan Glenn. Ailment: pneumonia
(69 )  2:09pm  Arrival: Ms Nevaeh Cruz. Ailment: cough
(69 )  2:09pm  Arrival: Mr Lincoln Le. Ailment: asthma
(69 )  2:09pm  Nathan Glenn is going to see Dr Jennings who'll be busy
               for 43 minutes, until (112)  2:52pm
               They have been waiting 0 minutes.

(70 )  2:10pm  Nevaeh Cruz is going to see Dr Ware who'll be busy for 6
               minutes, until (76 )  2:16pm
               They have been waiting 1 minutes.

(71 )  2:11pm  Arrival: Mr Leon Wiley. Ailment: food poisoning
(71 )  2:11pm  Arrival: Mr Louis Roman. Ailment: cold
(71 )  2:11pm  Lincoln Le is going to see Dr Gomez who'll be busy for 13
               minutes, until (84 )  2:24pm
               They have been waiting 2 minutes.
================================================================================
(75 )  2:15pm  0 doctors free, 2 patients waiting, Longest queue: 3,
               Longest wait: 2 minutes
          WAITING
          Leon Wiley
          Louis Roman
================================================================================
(76 )  2:16pm  Arrival: Mr Charles Acosta. Ailment: pneumonia
(76 )  2:16pm  Arrival: Mrs Sofia Villarreal. Ailment: pneumonia
(76 )  2:16pm  Arrival: Ms Charlotte Merrill. Ailment: food poisoning
(76 )  2:16pm  Finished: Dr Ware has finished treating Nevaeh Cruz
(76 )  2:16pm  Leon Wiley is going to see Dr Ware who'll be busy for 14
               minutes, until (90 )  2:30pm
               They have been waiting 5 minutes.

(84 )  2:24pm  Finished: Dr Gomez has finished treating Lincoln Le
(84 )  2:24pm  Louis Roman is going to see Dr Gomez who'll be busy for 12
               minutes, until (96 )  2:36pm
               They have been waiting 13 minutes.

(87 )  2:27pm  Arrival: Mr Ethan Henderson. Ailment: food poisoning
(87 )  2:27pm  Arrival: Mr Adam Rasmussen. Ailment: measles
(90 )  2:30pm  Finished: Dr Ware has finished treating Leon Wiley
================================================================================
(90 )  2:30pm  1 doctors free, 5 patients waiting, Longest queue: 5,
               Longest wait: 13 minutes
          WAITING
          Charles Acosta
          Sofia Villarreal
          Charlotte Merrill
          Ethan Henderson
          Adam Rasmussen
```

```
================================================================================
(90 )  2:30pm  Charles Acosta is going to see Dr Ware who'll be busy for
               29 minutes, until (119)  2:59pm
               They have been waiting 14 minutes.
```