FIT9136 Assignment 2

Last Update: 20/05/2020

Version: 1.5.1

Table of Contents

1. Introduction

This assignment is due on **Mon 8 June at 5:00 pm**. It is worth 30% of the total unit marks. A penalty of 20% per day will apply for late submission. Refer to the FIT9136 Unit Guide for the policy on extensions or special consideration. Any request for an extension must be made centrally.

Note that this is an individual assignment and must be your own work. Please pay attention to Section 4.2 of this document on the university policies for the Academic Integrity, Plagiarism and Collusion.

All the program file and any supporting documents should be compressed into one single `zip` file for submission.

(The submission details are given in Section 7.)

This document constitutes a **requirement specification** for you to develop your solution around, and you will be assessed on your ability to meet the specified requirements as closely as possible.

Allowed libraries

For this assignment, you may import the following libraries **only**, if you wish to do so. You are not expected to import *all* the libraries listed, but may choose the libraries you require from the following list:

- `math`
- `random`
- `numpy`
- `scipy`
- `matplotlib`
- `pandas`

Template and data files

Source file templates are provided to help you complete this assignment. You are expected to complete the assignment code within these files, by replacing placeholder lines with your own code and comments. In your submission, you should rename these files to replace any instances of xxxxxxxx with your student ID number.

Instead of copying code into later tasks, the different task files are linked together by `import` statements. You'll need to ensure all your `.py` files are part of the same project in your IDE (and modify these statements with your actual student ID) to ensure the Python interpreter can locate the other files in your assignment.

Compliance with specification

Do not rename the specified methods, change the number of arguments or return types. For example, if this specification states that a function should return a `string`, but you return a `list` instead, this may cost you marks. Likewise, note that function and method names in Python are case sensitive and must be spelled correctly. If we run a series tests on your code to check your implementation of a function named `some_function`, but you have defined `Some_Function` instead, it will appear that your function is missing when we test your submission against the specifications in this document.

Corrections and clarifications to this specification

A dedicated Assignment 2 forum will be provided on the FIT9136 Moodle site, with a **pinned thread** where updates and answers to frequently asked questions will be provided. Official replies will be posted every few days until the due date. This is an independent learning exercise. When asking a question on this forum, it is important that you provide details of what steps you have already taken to solve the issue independently, and provide a descriptive title so that other students can find the topic easily. Before starting a new forum topic, check to see if your question has already been asked in an existing discussion thread.

2. Background

In the past, there have been viral disease epidemics (including the 1918 influenza pandemic) which have got out of hand. We would like to simulate the way such diseases spread, to better understand how this happens.

In this assignment, you will create the necessary data structure to simulate social links and disease transmission between people, simulate infections among the population over a period of time, and plot graphs to determine whether an outbreak is contained or not.

The model we'll use is a simplification of the real world; we will not use actual parameters or consider all important factors. However, on completion of this assignment, you will have a better understanding of how such simulations are written in the real world.

The source data

We surveyed 200 fake people, all with unique names, and asked them to provide the names of their friends in the group who they are in regular contact with. You may assume that each person has specified at least one friend, and each of the person's friends has also named that person as a friend.

Please download the file named `a2_sample_set.txt` as your program will need to open and read data from this file. You may place the file in the same folder as your assignment code. [1]

The data file consists of 200 records. Each record is recorded on its own line in the file, and consists of a person's name, together with all the names of that person's friends. The real file is 200 lines long, but to illustrate the file format, just the first two lines of the file are shown here as a sample:

```
Gill Bates: Jodee Killam, Natacha Osterhoudt, Jom Tones, Verdie Tong, Ossie Digangi
Jom Tones: Marry Blakely, Masako Miguel, Gill Bates

. . .
```

Syntax of each record line: As seen in the example above, each line of data in the file consists of the first and last name of a particular person, followed by a colon and a space character "**:** " and then the first and last names of each of their friends (people they are in regular contact with) with each friend's name separated by a comma and a space "**,** ".

Example interpretation of source data above:

- The first line in the file is the record for Gill Bates. Gill Bates has named the following people as her social connections: Jodee Killam, Natacha Osterhoudt, Jom Tones, Verdie Tong, and Ossie Digangi. This means that if Gill Bates is *contagious* (able to spread the virus), Gill Bates may infect the people she has named, and if her friends are contagious, they may infect Gill Bates.

---

[1] Including this file with your submission is recommended but optional. This text file does not contain any special Unicode characters so it does not need to be opened with any special encoding.
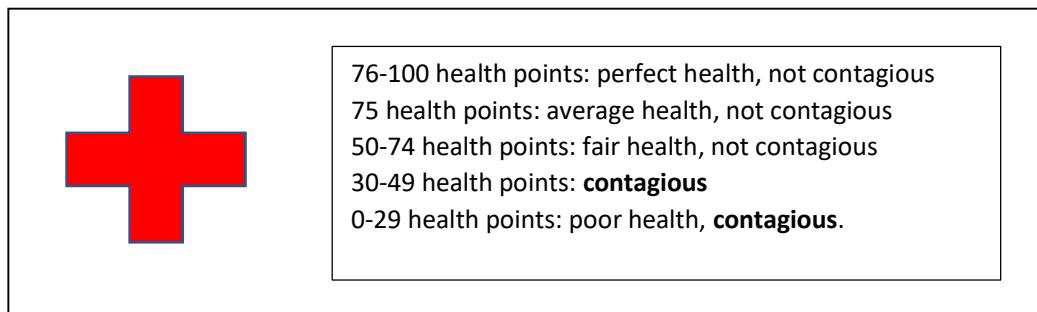
- On the next line, Jom Tones has named his friends in a similar way, and so on.
- Note that Gill Bates has named Jom Tones as one of her friends. This means that Jom Tones must also name Gill Bates as one of his friends. It's not unusual that may both visit each other, and the virus may travel from either person to the other. You can assume that this rule is followed for all records in the file.

How the virus spreads

Health of each person

Each person has a number of **health points** which changes over time depending on a person's health.

The government has published the following guidance about health points. The number of health points is used to check if a patient is contagious (i.e. able to infect other people) or not:

76-100 health points: perfect health, not contagious
75 health points: average health, not contagious
50-74 health points: fair health, not contagious
30-49 health points: **contagious**
0-29 health points: poor health, **contagious**.

When sleeping after each day, each person's immune system will naturally add 5 health points to that person, up to a maximum of 100 health points.

Health points are not required to be in an integer form, so each time you need to check if a person is contagious or not, the person's current health should be rounded to the **nearest** integer. The maximum possible health of a healthy person is 100 (the health point value of each person should be limited to this maximum), and the minimum possible value is 0. A person with 0 health can still recover, but cannot lose any further health points.

Meeting probability

Each day, a person *may or may not* **visit** another person for a meeting. For each person, the probability that they will travel to visit one of their friends depends on social distancing regulations. A single **meeting probability** parameter will be applied to all people in the group to determine the effect of a certain level of social distancing. This probability is a fraction of 1. For example, running the simulation with a meeting probability of 1.0 means that every day, every person will leave home to visit *all* of their friends, and all their friends will *also* travel to visit them during the same day. A probability of 0.0 means nobody can leave home to visit anyone else, and a probability of 0.333 means there is a 33.3% random chance of each visit happening.

Viral load

The virus spreads when a **contagious** person[2] passes a ***viral load*** to a person they are visiting, or a person who has visited them. The term 'viral load' is a measure of the quantity of virus in the air which the other person breathes in when they are visiting and/or being visited by any contagious person. A person can be affected by a viral load even if they are already partly sick.

The viral load produced by a contagious person is given by the following formula (where L_v is the viral load produced, and HP_c is the number of health points of the contagious person who spreads the virus):

---

[2] This means a person whose current health point value is in one of the ranges listed as **contagious** according to government guidance. A non-contagious (healthy) person does not produce a viral load at all.

$$L_v = 5 + \frac{(HP_c - 25)^2}{62}$$

A small viral load will not make a healthy patient sick, but a larger viral load or viral loads from several people might reduce a patient's health to the point that they become contagious with the disease and begin to spread it to others. Also, our L_v formula shows that a contagious person who is only mildly sick may produce a larger viral load than a person whose health has worsened, due to the nature of this disease.

Effect of infection

When a contagious person produces a viral load, every person they meet when visiting (or being visited) will be infected by their viral load. If the viral load is small, or a person is healthy, the person who is infected might not become sick, and they will quickly recover their health later when they sleep.

The change in health from **receiving** a viral load from another person is given by the following formula (where HP_a is the current health points of the recipient **before** the viral load hits them, and HP_b is the new value of person's health points **after** receiving the viral load). The formula is different depending on the health of the person who receives the viral load:

$$HP_b = \begin{cases} HP_a - (0.1 \times L_v), & \text{if } HP_a \leq 29 \\ HP_a - (1.0 \times L_v), & \text{if } 29 < HP_a < 50 \\ HP_a - (2.0 \times L_v), & \text{if } 50 \leq HP_a \end{cases}$$

Now, your assignment…

3. Task 1: Representing social connections in your program

You should write your code for this task within the a2-xxxxxxxx-task1.py template file provided (rename this file to replace xxxxxxxx with your student ID).

Person class

Create a Python program that defines a `Person` class with the following methods.:

- `__init__(first_name, last_name)`
  - \# Constructor method to create a new Person object.
  - \# where `first_name` and `last_name` are strings containing the person's name
- `add_friend(friend_person)`
  - \# This method should add a new social connection to be stored in this `Person` object.
  - \# `friend_person` is a reference to *another* Person object.
- `get_name()`
  - \# This method returns a **string** containing the person's first and last name concatenated together; e.g. "`Jom Tones`"
- `get_friends()`
  - \# This method returns a **list** of `Person` objects for the social connections that have been added.

The purpose of the `Person` class is to represent a person who is linked to other people by social connections. The above description specifies what methods and form the `Person` class required to have, but not how to code them.

Therefore each `Person` object must be able to store a set of references to other Person objects. (You will need to think about a suitable data type for storing a set of objects, and how you would need to initialize the empty set within the class definition.) In addition to storing the person's friends, your `Person` class should also contain instance variables to keep track of the person's name. If you wish to do so, you can extend this class by adding more methods as needed to help you complete the rest of the assignment.

load_people function

After implementing this class, implement a separate function in your program named `load_people()` which does the following:

1. Opens the file `a2_sample_set.txt` which contains the data set given.
2. Creates a new Person object for each record (line) in the file, which contains the name of the person represented by that record.
3. Where a person's record indicates that they have friends, you should use the `addFriend()` method to add each friend to that Person object. [3]
4. Finally, return a `list` of all the `Person` objects that have been created from the file records (making sure to have closed the file first).

4. Task 2: Simulate disease spread

You should write your code for this task within the `a2-xxxxxxxx-task2.py` template file provided (rename this file to replace `xxxxxxxx` with your student ID).

Patient class (a subclass of *Person*)

The Person class you defined in task 1 is fine for mapping social connections, but it does not contain appropriate methods for simulating disease spread or health of the people in our group.

Before you move to writing the simulation (later in this task), define a `Patient` class which inherits the methods of the Person class through **inheritance**. Your `Patient` class should add the following methods:

- `__init__(first_name, last_name, health)`
    - \# Constructor method to create a new `Patient` object by inheriting from the `Person` class.
    - \# where `first_name` and `last_name` are strings containing the person's name
    - \# and `health` is the initial starting value of the person's health points.
- `get_health()`
    - \# This method returns the patient's current health points. [4]
- `set_health(new_health)`
    - \# This method changes the patient's current health points directly.
- `is_contagious()`
    - \# This method should return a Boolean result of `True` if the person's health points are in the range of being contagious (able to spread disease) as defined in section 2. It should return `False` if the person is not currently contagious.
- `infect(viral_load)`
    - \# This method infects the `Patient` object with a viral load. It causes the patient to receive the viral load specified in the method's argument (given as a floating point number).
    - \# After receiving a viral load, the person may or may not become sick enough to be contagious.
    - \# Calling this method should adjust the person's health points according to the rules defined in section 2. Note that the person's health cannot go below 0 as discussed in section 2.
- `sleep()`
    - \# Calling this method causes the person to recover some health points (one night's sleep) according to the rule defined in section 2.

---

[3] Caution: Each different `Person` should be represented as one unique object within the simulation. Do not create a new `Person` object each time you add a person as a friend, otherwise your simulation will have different people with the same name who are not linked together.

[4] Make your own decision as to whether you prefer to return an integer or floating point value here. If returning an integer value, make sure to round to the nearest integer. This means that 45.50 is rounded up to 46, but 45.49 is rounded down to 45.

To implement these methods, your `Patient class` should also contain an instance variable to keep track of the person's current health points.[5] If you wish to do so, you can extend this class further by adding more methods as needed to help you complete the rest of the assignment.

load_patients function (a new version of the *load_people* function in task 1)

You will also need to write a slightly different version of the function you wrote to load data from the file, since we really need a list of `Patient` objects, not a list of `Person` objects. Write a function named `load_patients(default_health)` which does the following:[6]

1. Reads from the file `a2_sample_set.txt`.
2. Creates a new `Patient object` for each record (line) in the file, which contains the name of the person represented by that record. *For each Patient object created, you should assign the health value given by the default_health argument, since the initial health of each person is not listed in the file.*
3. Where a person's record indicates that they have friends, you should use the inherited `add_friend` method to add each friend to that Person object.
4. Finally, return a `list` of all the `Patient` objects that have been created from the file records.

In other words, this function should do the same thing as `load_people`, except that it should create `Patient` objects (with a specified default health value) instead of `Person` objects.

run_simulation function – *implement the simulation logic…*

Now implement a `run_simulation(days, meeting_probability, patient_zero_health)` function which should implement the following behaviour:

1. Take in the following arguments:
   a. `days`: the integer number of days the simulation should run for.
   b. `meeting_probability`: the fractional probability that any person may visit a certain friend on a certain day. 0.0 means 0% chance, 1.0 means 100% chance. (This was explained in section 2.)
   c. `patient_zero_health`: the initial health of the first person in the file. (See below.) If the (rounded) initial health is less than or equal to 49, this person is contagious and there may be the chance of a disease outbreak.
2. Use your `load_patients` function to load patient data from the disk. The first patient in the returned list (who we will call 'patient zero') should be given the starting health value specified in the `patient_zero_health` argument. The remaining patients should be given an initial health value of 75, which is the average health of the population from section 2.
3. Run through each day of the simulation. For each day, do the following:
   a. For each patient in the group, look at each of the person's friends, and randomize whether the person should meet that friend today.[7] The probability for this to happen is given by the `meeting_probability` argument. If the meeting takes place, each person in that pair who is contagious[8] should spread a viral load to the other person, by calling the `infect()` method on the other person. You will need to calculate what viral load to infect each friend with according to the rules in section 2.
   b. After all meetings have completed for the day, check how many people are now contagious in the simulation.

---

[5] Alternatively, this instance variable could be provided in the `Person` base class if you deem more appropriate.

[6] You may copy and adapt the code you use to write the `load_people` function in task 1. You may also call any helper functions from your task 1 file as appropriate.

[7] Hint: the week 3 lab contained an exercise using the `random.random() function`.

[8] This could be checked at the beginning of the meeting or during the meeting, as long as you make some sensible design decision here. Once a patient becomes contagious, they become able to infect others in their remaining interactions from that point onwards, including any later meetings in that day.

c. After the end of each day, all people should `sleep()` to recover some health.

4. Finally, the function should return a list with the daily number of contagious cases through the duration of the simulation. (as measured in 3.b.) For example, if your simulation runs for 30 days, the list should contain 30 integers containing the number of people who were contagious at the end of each day, from the first day of the simulation (element [0]) to the last day of the simulation (element [29]).

You may write your own code within the __main__ block to test your simulation. Note that due to random probability, you may get a different result each time you run your simulation, depending on your parameters.

5. Task 3: Visualise the curve

Complete this task in the a2_xxxxxxxx_task3.py template file (change xxxxxxxx to your student ID).

visual_curve function

Write a function named `visual_curve(days, meeting_probability, patient_zero_health)` which runs the simulation using the specified arguments, and then does the following:

1. Runs the simulation by calling the `run_simulation` function with the specified arguments for `days, meeting_probability, patient_zero_health`.
2. After the simulation is complete, prints the whole daily list of contagious patient counts from the returned data.
3. Then, using functionality from either the `matplotlib` or pandas library, plot a curve showing the daily number of contagious patients over the number of days of the simulation. The days of the simulation should be on the X axis and the contagious patient count on the Y axis. Your graph should have the X and Y axis labelled accordingly.

main block code for running the program

To test your code easily, add code to the main block of the file to **ask the user to input** the **number of days** to run the simulation, the **meeting probability** to use for the simulation, and the **health of Patient Zero** in the simulation. Your program should then call the `visual_curve` function to produce a result based on the values entered. You should prompt the user for these values each time the user runs your Task 3 program. [9]

Testing your simulation

In your output graphs produced by your simulation, the rate at which the curve rising indicates how quickly the virus is spreading, and a flat or decreasing curve means the virus is no longer spreading. Some sample plots are shown here:


(a)

(b)
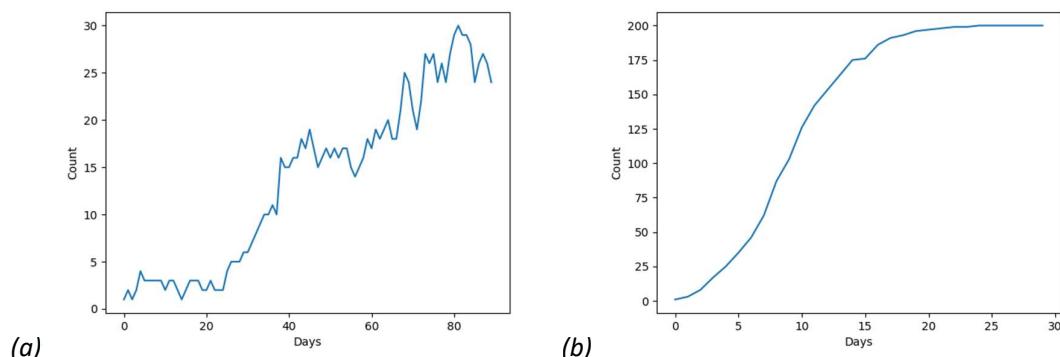
Figure 1: Sample curve plots showing (a) a slow rate of spread to 30 people and (b) a fast rate of spread to all 200 people.

Now that you have finished coding, try to run your program with the given parameters for the following scenarios. For each test case, save the resulting visual graph as a `.png` image file after running your program and include this

---

[9] You may assume the user will provide valid input and are not required to carry out input validation here.

FIT9136 Assignment 2 Specification - Page | 8

with your submission. Name these files as `scenario_A.png`, `scenario_B.png` and `scenario_c.png` respectively.

Test scenario A: an uncontained outbreak

Prediction: In this scenario, there are many social gatherings, and 'patient zero' is quite unwell at the beginning of the simulation. The situation might get out of control very quickly.

- Number of days: 30
- Meeting probability: `0.6`
- Patient zero health: 25 health points

Test scenario B: an unpredictable situation

Prediction: In this scenario, there are minimal social distancing restrictions implemented (to discourage people from meeting), and 'patient zero' only has very mild symptoms but is maximally contagious. In this scenario the situation could be very unpredictable and depend on pure luck. You may find you get a different result each time the simulation runs based on random probability. Sometimes patient zero may not spread the disease at all before recovering and there is no outbreak. In unlucky cases, the virus may begin to spread.

- Number of days: 60
- Meeting probability: `0.25`
- Patient zero health: 49 health points

Test scenario C: flattening the curve

Prediction: In this scenario, social distancing restrictions on meetings could mean that while there is the chance for an outbreak, it is more likely that the disease will die out quickly, case numbers will flatten out, or only grow slowly.

- Number of days: 90
- Meeting probability: `0.18`
- Patient zero health: 40 health points

Remember that your simulation is based on random probability. 200 people is a very small sample size to predict a real disease pandemic, and you may find that your results are very different each time you run your program. This is normal. Do the results match the predictions? Try to explain (very briefly!) any reasons why or why not within your task 3 file header comments.

6. Important Notes

Documentation

Commenting your code is essential as part of the assessment criteria.

You should include file header comments at the beginning of each program file, which specify your name, your student ID, the start date, and the last modified date of the program, as well as with a high-level description of the program. Provide function header comments at the top of each function/method to explain the purpose, arguments and return value of each function. In-line comments within the program (to explain the logic within a function) are also important and required.

Marking Criteria

Each of the three tasks in this assignment carries an equal weight. For each task, you will be graded based on the following marking criteria:

- 50% for working program - functionality;
- 15% for code architecture - algorithms, data types, control structures, and use of libraries;

- 15% for coding style - clear logic, clarity in variable names, and readability;
- 20% for documentation - program comments.

Auto-marking may be utilized, which means that deviating from the stated requirements may result in the auto-marking algorithm being able to call the specified functions or methods, or check the expected return values.

As an assessment task, you cannot ask tutors to check your code or tell you if it is correct. If you need to clarify a coding concept or technique, you may ask questions in general terms and only using code examples which do **not** relate to the assignment requirements. Non-coding questions related to interpretation of the specification may be directed to the teaching team in consultations and labs, as well as the dedicated Assignment 2 forum.

7 Submission

You are required to submit your assignment to Moodle as a .**zip** file named with your Student ID.

For example, if your Student ID is 31303030, you would submit a zipped file named "A2_31303030.**zip**".

Note that naming your submission file correctly is an important part of demonstrating comprehensions of this assignment specification. Marks will be deducted if this or other specified requirements are not complied with (For example, uploading an .rar file when a .zip is specified).

Your submission must be done via the assignment submission link on the FIT9136 S1 2020 Moodle site by the deadline specified in Section 1. Submissions will not be accepted if left in Draft mode. Be sure to submit your files before the deadline to not incur late penalties.

7.1 Deliverables

Your zipped submission should contain the following documents:

Three Python scripts named as follows (based on the template files provided):

- `a2_xxxxxxxx_task1.py`
- `a2_xxxxxxxx_task2.py`
- `a2_xxxxxxxx_task3.py`
- Images of graphs to support your task 3 findings.

Again, you should replace xxxxxxxx with your student ID.

Note: Your programs must run in the Python 3 environment specified at the start of semester. Submit all files in a .**zip** file named as described above. Do not submit a .**rar** or other format. Marks may be deducted for any of these requirements that are not strictly complied with.

7.2 Academic Integrity: Plagiarism and Collusion

Plagiarism - Plagiarism means to take and use another person's ideas and or manner of expressing them and to pass them off as your own by failing to give appropriate acknowledgement.

This includes materials sourced from the Internet, staff, other students, and from published and unpublished works.

Collusion - Collusion means unauthorised collaboration on assessable work (written, oral, or practical) with other people. This occurs when you present group work as your own or as the work of another person. Collusion may be with another Monash student or with people or students external to the University. This applies to work assessed by Monash or another university.

Automated plagiarism checking will be used to investigate similarity in blocks of code beyond random chance.

It is your responsibility to make yourself familiar with the University's policies and procedures in the event of suspected breaches of academic integrity. (Note: Students will be asked to attend an interview should such a situation is detected.) The University's policies are available at:
http://www.monash.edu/students/academic/policies/academic-integrity.