COMP7230 – Introduction to Programming for Data Scientists
Assignment 2 Due Sunday 6 September 2020, 11:59pm

Last update 21 August 2020

Overview and Objectives

This assignment assesses some of the topics we have covered more recently in the course. There is a reduced emphasis on actual coding and a greater emphasis on understanding, using and modifying existing code in order to solve a particular problem.

You are given a (more-or-less) working piece of software, that simulates a pandemic outbreak in Australia. The pandemic could be anything from the current COVID-19 outbreak to a zombie apocalypse. A similar approach could be used to model memes spreading out across the internet, news spreading over Twitter or animals spreading in a new habitat.

This software is much more complex than what you are expected to be able to create on your own in this course, however it is a good example of the type of thing you might be able to find on the Internet and use to solve a problem. As such, your tasks are to refactor and modify the software, to improve the documentation and add logging functionality. Finally, you are asked to use the software to answer two questions about where to target a response to the pandemic.

Please also note, that while this software is (hopefully) useful for education purposes, it isn't actually a particularly good model of the current pandemic. It is done at the wrong resolution and doesn't consider many factors such as lock-downs, international arrivals and so forth.

Important

- Make sure that your student ID is included as a comment at the start of your submission.
- Submit **one Python file only**, named `COMP7230_Assignment_2_Submission.py`.
- Make sure you submit your assignment before the submission deadline.

Submission

Submission will be done using Wattle. Click on the link Assignment 2 submission (to be made available) in the assessment section of Wattle to upload your file. You may submit as many versions of the assignment as you wish. We will mark the version present on the due date, or the first submission after the due date if there is no submission on the due date.

Deadlines, Extensions and Late Submissions

The assignment is due by 23:59 (11:59pm) on Sunday, September 6, 2020.

Students will only be granted an extension on the submission deadline in extenuating circumstances, as defined by official ANU policy. If you think you have grounds for an extension, you should notify the course convener as soon as possible and provide written evidence in support of your case (such as a medical certificate). The course convener will then decide whether to grant an extension and inform you as soon as practical.

Late submissions without extension will incur a 5% penalty per business day. No submissions will be accepted more than two weeks late or after marks have been released. Given the need to finalise the course marks in order to process enrolment in the second half of the GCDE program, we aim to have the marks returned well before the two week deadline.

Once marks are released, you will have two weeks in which to question your mark. After this period has elapsed, your mark will be considered final and no further changes will be made. If you ask for a re-mark, your assignment will be re-marked entirely, and your mark may go UP or DOWN as a result.

Plagiarism

No group work is permitted for the assignment. We do encourage you to discuss your work, but we expect you to do the assignment work by yourself. If you are unsure about what constitutes plagiarism, please read through the ANU Academic Honesty Policy.

If you do include ideas or material from other sources, then you clearly have to make attribution. For example, by providing a reference to the material or source as a comment in your code. We do not require a specific referencing format, as long as you are consistent and your references allow us to find the source, should we need to while we are marking your assignment.

Any student may be asked to explain any part of their submission and if unable to satisfactorily do so, marks may be adjusted or further action taken in accordance with the ANU academic honesty policy.

Assignment Structure

The assignment consists of two python files:

- `COMP7230_Assignment_2_Submission.py`
- `COMP7230_Assignment_2_Submission_Tests.py`

and the following helper files:

- *final_city_data.csv*
- *Aus_Map.PNG*

You should download all these files to the same location, before starting to answer the assignment questions.

Simulation Basics

This section provides a basic overview of the approach that the software takes to simulating a pandemic.

Australia (or any other region if the data and map files are changed) is modeled as a graph. Major population centres are captured as instances of the `City` class. Connections between cities (roads, railways, etc.) are captured as neighbours on the `City` class, in other words, if two cities are connected, they will store each other as neighbours. If you wish to see the connections on the map during the simulation, you can uncomment the marked section of the `animate_map` function to display them.

Once the data is read in and the graph is created, the simulation sets some parameters based on the chosen value of `SIMULATION_NUMBER`. These control thing like the mortality rate (i.e. the proportion of people who get the disease who will die), the duration of the illness (i.e. how long it takes after contracting the disease for someone to die or recover), etc. The simulation number also controls things like where the infection starts from and whether there are any treatment centres present. For example, simulation 0 (the default) starts the pandemic in *Alice Springs* while simulations 1 and 2 start the pandemic in *Rockhampton*, *Brisbane* and the *Gold Coast*. You can change the simulation number as you wish to look at how the pandemic might proceed with different starting locations and parameters.

Once the simulation is setup, it proceeds in a sequence of turns, or steps. Each turn the following happens:

- The `start_of_turn` method is called for each city, which moves any infected cases on route to the city, into the city proper.
- The `run_turn` method is called on each city. This sends some proportion of the infected cases in the city to the neighbouring cities (though they won't arrive until the following turn). Then some proportion of the infected cases in the city either die or recover. Finally, any remaining infected cases spread the disease to healthy inhabitants.
- The `run_turn` method is called on each treatment centre. The treatment centres cure any infected cases in the city they are in (as long as their supply of cure is sufficient).

The main work of the simulation is done in the `run_turn` method on the `City` class, which is responsible for moving the infected cases around the map and generally controlling the spread of the pandemic. Once a turn is complete, it is displayed visually on the map along with some statistics. Then, the next turn begins and the whole process repeats, until either there are no infected cases left, or a pre-defined stopping condition is reached.

Assignment Tasks

You only need to modify and submit `COMP7230_Assignment_2_Submission.py`. If you wish to add your own unit-tests, you can include them in `COMP7230_Assignment_2_Submission.py`, and use them to check your work. However, they will not be marked.

In addition to the file `COMP7230_Assignment_2_Submission.py`, we have also provided a suite of unit tests, `COMP7230_Assignment_2_Submission_Tests.py`, which will help you to test your work. You will find this particularly helpful for doing the refactoring in Question 1, where at any point in your refactoring, you should still be able to run and pass the unit-tests. However, as always, please note that the tests are there to assist you, but passing the tests is NOT a guarantee that your solution is correct.

The Assignment consists of 3 parts and 5 questions broken down as follows:

Part 1: 20 Marks

For Part 1, you only need to modify the `City` class, though you may find it helpful to have a look through other parts of the code in order to see how it works. However, if the 'bigger picture' is proving too difficult, focus on the code in this class definition and the logic behind each specific method. Understanding how this class works will likely be at least 75% of the battle for Part 1.

- Question 1: Refactor the `run_turn` method on the `City` class. 8 marks.

  Separate the code into smaller, simpler methods for better understanding and ease of testing. You can use the three methods provided (`move_infected`, `change_in_infected_numbers` and `spread_infection`) or create your own as you see fit. You can also modify the method signatures/return values as you wish. The `run_turn` method should then just call your new methods as appropriate. At any point in the refactoring, you should still be able to run the unit-tests for the `run_turn` method. This is a good example of the power of unit-tests, since it helps you check nothing has broken through the revision and reorganisation of your code.

- Question 2: Setup a logging process. 6 marks.

  To begin with, the simulation displays a lot of graphical information to the user, however, once the simulation is over, this information is lost. As a result, you need to write some valuable information to a text file so that it can be preserved for subsequent analysis. At a minimum, you should log the following significant events, and the turn number on which they occurred:

  - The pandemic reaching a city for the first time (first infected case in a city).
  - A city becoming infection free (can happen more than once per city per simulation).
  - Everyone in a city becoming infected.
  - Everyone in a city being dead.

  You can also choose to log additional information if you wish, but keep in mind that too much detail in a log file is almost as bad as too little.

- Improve the code documentation and naming of the `City` class. 6 marks.

  The current variable naming, commenting and code structure in the `City` class is terrible. Perhaps the author was rushing to finish it, or gave up on the project. Whatever the reason, it needs to be significantly improved to aid in understanding and testing. In addition to the coding tasks in Questions 1 and 2, you should improve the overall documentation and naming of the `City` class definition. This should also include proper documentation for any of the methods you created for Questions 1 and 2. You may have to research what some of the existing methods do in order to properly document them. If you change the names of any variables, please make sure to run both the unit-tests and simulation to make sure nothing broke elsewhere in the code. You may have to choose between keeping a bad variable name and a wide scale refactoring. Be sure to document the reasons for your decision in such cases.

Part 2: 6 Marks

Often when you are using other people's code, you may find it doesn't do everything you want it to do. Being able to extend or modify code to introduce new functionality is an important skill. In this case, we would like it so that our treatment centres could move to areas where the pandemic outbreak is worst in order to more quickly stamp it out. Your task is to add this functionality to the `TreatmentCentre` class.

- Question 3: Write the `move` method on the `TreatmentCentre` class.

  While the simulation has a class to store treatment facilities (e.g. a medical centre which can distribute the cure), the current implementation makes these stationary. Your task is to write the code that makes them mobile. In each turn, they should look at neighbouring cities of the one they are currently located in and move to the city with the highest number of infected cases. If the highest number of infected cases are in their current city, they should not move at all.

Part 3: 10 Marks

The final part is challenging, so make sure you have a good solution to Parts 1 and 2 before spending a lot of time on these questions.

In this part, we are now going to look at how you might use the software in order to answer questions. In this case you are being placed in the role of someone coordinating the response to the pandemic, and you have to decide how to allocate resources in order to minimise the impact of the outbreak.

You can modify **any** part of the code in order to answer these questions, however, the bulk of your work should be done in the indicated part of `__main__`. If you modify code outside this location, please put it into an if/else block, i.e.:

```
if SIMULATION_NUMBER == 4:
   do stuff
```

This will stop it breaking your answer to other questions.

- Question 4: Road Closures. 5 Marks

  Make sure to set `SIMULATION_NUMBER` to 4 when working on this question. Assume that after 20 turns, a massive shipment of medicine is going to arrive and that you could cure anyone still infected at that point. Also assume that you can use the defence force to close off 3 connections on the map until then. Which 3 should you close in order to minimise the total loss of life? The code you write should show how you got your answer. Make sure it is well commented.

- Question 5: Medicine Distribution. 5 Marks

  Make sure to set `SIMULATION_NUMBER` to 5 when working on this question. Assume that instead of a treatment centre having a massive stockpile of medicine, the medicine becomes available in small allocations which you can distribute each turn. Each turn, you get to add a new treatment centre (with medicine to cure 1000 people), at a city of your choice. What is the best allocation sequence to minimise the total number of people who get infected prior to the end of turn 20. As with Q4, your code should demonstrate how you got your answer. Make sure it is well commented.

Code Quality: 4 Marks

There are 6 marks in Part 1 allocated to improving the quality of the code in the `City` class, along with any additions you make to the code at that point.

In addition to that, your code for Parts 2 and 3, will be assessed for commenting, variable naming, efficiency, organisation and other quality measures, and this will be worth up to 4 marks.

Marking

The assignment will be marked out of 40 and will count for 40% of your final grade for COMP7230. Partial marks may be awarded, even for solutions that do not pass all the tests, as long as the code is making progress towards a correct solution and it is well structured and well commented. As such you are encouraged to attempt questions even if you feel you cannot get a complete or perfect solution.