Documentation for GEOG5995M Programming for Social Scientist: Core Skills, Assignment 2:

**Planning for Drunks.**

**December 2022**

The following report documents the process of development for the project ‘Planning for Drunks’ including issues, limitations, and solutions throughout the creation of the code for the project. Sources can be found at the end of this document.

**Show me the way to go home…**

The code submitted for assignment 2 is designed to read a data file, present home and pub points, then create 25 drunks, all of which are numbered and randomly move around the map. They do not retrace their steps and stop when they reach their assigned home. Furthermore, the code produces a csv file which presents the density of the drunks throughout the map.

The following section describes the development process of the code, starting with my first ideas through to the finished code.

**First Ideas**

The following brief was given for the project idea, broken down into 5 steps. For each step, I have created my initial ideas and thoughts.

1. Pull in data file, find home and pub points.

* Download file as txt
* Read file using CSV library
* Using a for loop, read document line by line and save to list

1. Draw pubs and homes on the screen.

* Using matplotlib library plot environment onto a figure

1. Models drunks leaving pub and reaching home and stores how many drunks pass through each point.

* Create drunks as agents
* Give them random movement
* Unsure how to deal with passing through each point

1. Draws density of drunks passing through each point

* Create second plot for density
* Use matplotlib library

1. Saves density map to a file as text.

* Save as file using CSV library

As can be seen from my initial ideas, at the commencement of the development process I had significant ideas on how to deal with the file, create the drunks, and the map. However, I had difficulties in understanding how to deal with the density map and how to make the drunks understand where their homes were and what they do when they reach it. To begin the project, I started to write the code for the areas which I felt confident, and later research the areas which I was less confident with, which led to breaking down the development of my code into 3 distinct steps.

**Step 1: Creating Environment, Drunks, and Animations**

Firstly, I wrote code to read the 300 x 300 raster file, loop through the file, and store the contents of the file in a list. Meanwhile, for the creation of the Drunks, I decided to use an agent framework with a class ‘Drunks’. Using an agent framework allows me to create simpler, more readable code. This meant for both the agents and environment I had to use the framework. Henceforth, using the self.\_\_init\_\_ section of the class, I connected the two files of code. I wrote simple code for the drunks under a move function with the class which randomly moved each agent from their starting point, the pub in the centre of the map.

Following this, I decided the best way to visualise this project would be through the use of an animation. To do this, I drew on what I had learnt from the module practicals and, using the matplotlib animation library, I created an animation to visualise the model.

At the end of the first section of development, I had created two files of code which mapped the input file onto a figure, creating a map of homes and pubs, created drunks which move randomly on the map starting from the pub, and coded an animation which visualised the model.

**Step 2: Numbering Drunks and Making them Smarter**

Through my development process in step 1, I had created drunks and houses; however, the drunks did not know which house belonged to them. To solve this, the drunks had to be numbered. Henceforth, I created a variable ‘self.house\_ids’ in the agent framework, which I then used in the main code to assign numbers 10-250 to each drunk using a for loop. I attempted to assign numbers using a while loop first, which did not work, and then I attempted to assign numbers within the agent framework, which also provided too many complications.

Secondly, now each drunk had their house numbers, I wrote code using 2 for loops and an if statement, which checks after each movement if the house number of the drunk is equal to the environment number it is on. If it is, the drunk stops moving as he is at home, if not the drunk continues to move.

The next issue I faced is the model was taking a very long time to run, because agents were allowed to retrace their steps. Hence, I decided to make the drunks a little smarter. To do this, I recoded the move function in the agent framework. I created a list for the both the x and y coordinates which remembers a set number of previous coordinates where the drunks have been. Then, when the random function yields a new coordinate, the code checks the list to see whether the drunk has been there in the last set number of steps. If it has not been there, the drunk moves to the coordinates, but if it has then the process repeats. Lastly, the new coordinate is added to the list and the first coordinate is deleted. The calibration of the length of the memory of the drunks created some issues, as too high a number would lead to drunks remembering all of their moves and leading to stalling, while too few had too significant an effect. Henceforth, the length of 15 was chosen after testing. To further develop the movement, I added a random integer generator to generate the number of steps each movement will take each agent.

This section could have been further developed by combining the X and Y coordinate memory lists, creating a function which directs each agent towards their house, or even the implementation of a function where, with each step, the drunks become smarter and smarter, modelling them sobering up.

**Step 3: Density Map**

The third step of the process was the area where I had to do the most research. I realised that, essentially, I had to make a copy of the environment where all values were 0, then add 1 to each point where the drunks pass through. Therefore, I created code which copies the dimensions of the input file, and creates a same size map with all 0’s. Then I added a line of code below the movement function call, which adds one to the coordinate where the agent is, hence creating a density map file. Next, I added a second figure for the output, which displayed the density of drunks in the town. This was an especially difficult process, as it took a lot of tweaking and analysing my code for the subplots function to work.

Finally, using the CSV library, the code saves the density map as a csv file which is updated with every movement. There is a known issue with the code at this point which is that the file is opened and updated with every move rather than copying in the final density map at the end. This was an issue which I was not able to resolve. However, the code does produce an accurate csv file showing the density of drunks.

**Step 4: Finishing Up**

The last edition to my model was to create an input choice for the user. If the user inputs 1, the output is a running model showing drunks leaving the pub and going home. As this model takes a while to finish, the option of 2 presents a finished output and density map.

To finish my code, I made sure the code was legible, comments were thorough and explanatory, and lastly that evidence of testing was shown. Testing for the code was continuous throughout the process. For a majority of the time, the testing was visual of the running model. Where evidence of testing was applicable, it has been provided and explained with comments. As for the testing and development of the move function, testing is shown in a separate document. Overall, the code works well and meets the brief and extension. The code is limited in that it is a little slow and could be more efficient.

**Sources**

The code produced for this project is all my own.

The project idea and input file was provided by the module coordinator in the form of 300 x 300 raster file available at: <https://www.geog.leeds.ac.uk/courses/computing/study/core-python-phd/assessment2/drunk.html>.

Throughout the project I have used a number of websites for assistance with my code, mainly for debugging including but not limited to: Python.org, Programiz.com, edureka.co.uk, stackoverlow.com.