Agent-Based Model Documentation

(word count: 702)

This document aims to guide the Agent-Based Model Program users with more detailed instructions and guidance pictures. This also acts as validation evidence while running the program.

This program consists of two source codes in the same directory as this documentation, which are model.py and agentframework.py. Prior to following this documentation, users should refer to the README.md file for installation, setup, and brief program introduction.

Model Initiation

Once users open the Python files in Spyder, the window appears as follows. Then, hit the "Run" button:

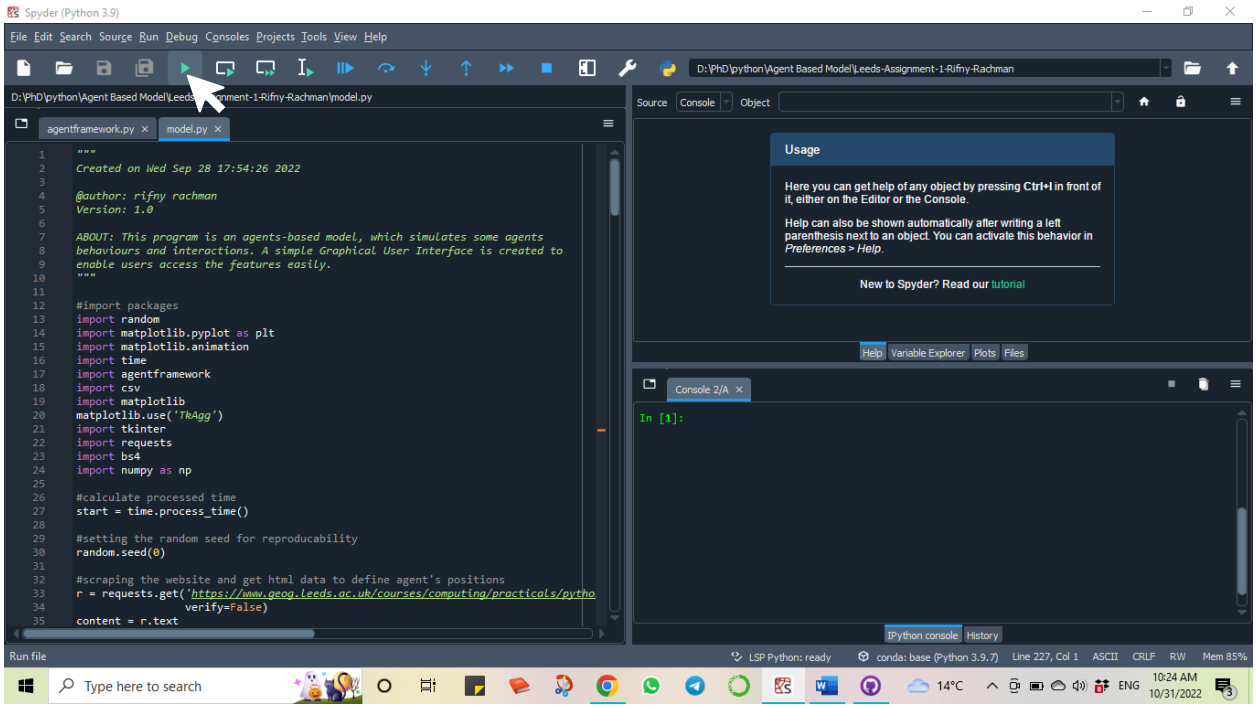

Figure 1 Run Model

It will pop up two windows, which are "Figure 1" and the GUI window titled "Welcome to Agent Based Modelling! ^^". Users shall focus on the latter and ignore the former. Meanwhile, on the console window, users could check the initial setup of the model.

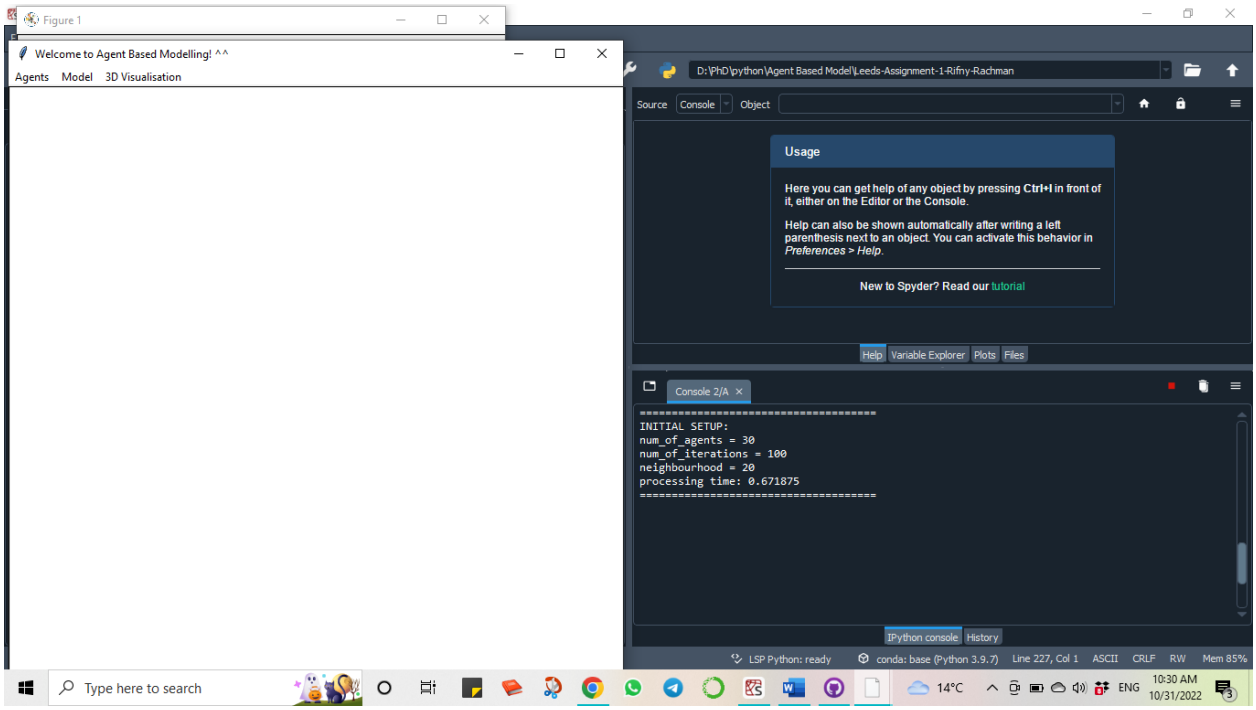

Figure 2 Initial Setup and Pop-Up Windows

Running the Model

In running the model there are several features as mentioned in the README.md file. In this documentation, each feature would be run, and the output would be shown, as follows:

✓ Agents → Status

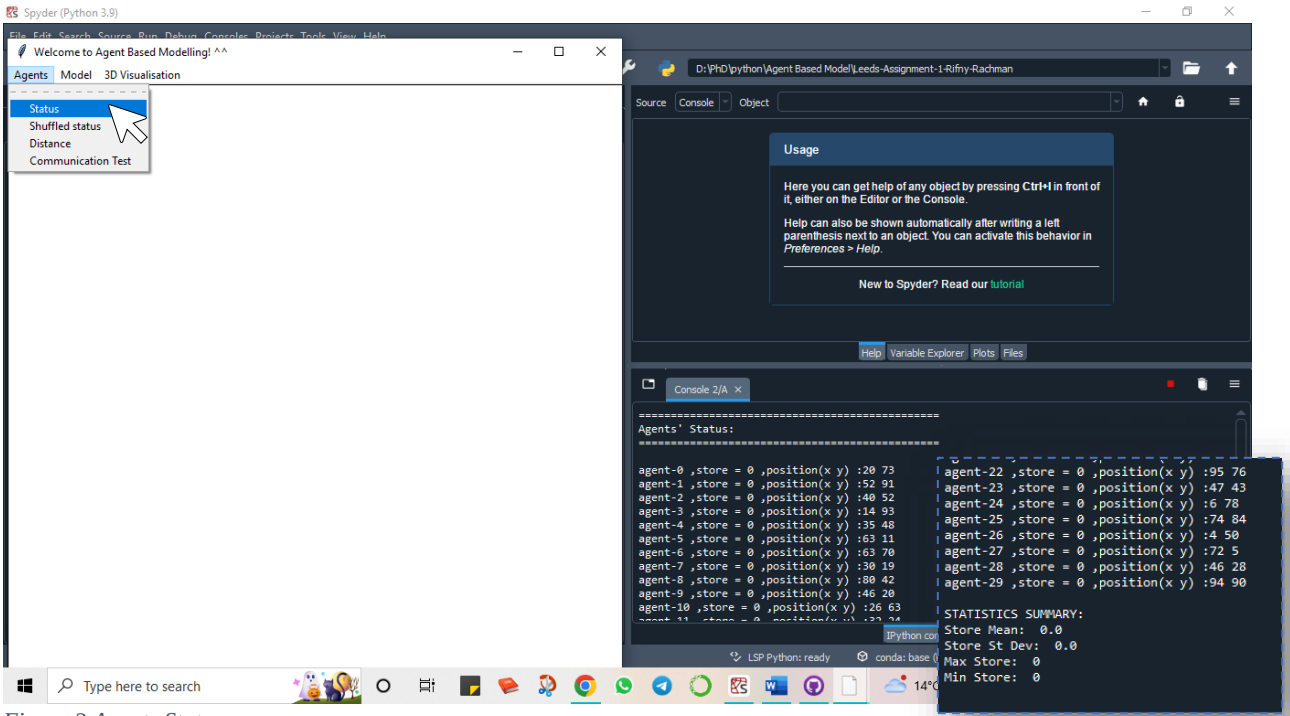

Figure 3 Agents Status

Agents Status feature yields updated information including each agent's store and position. At the end of the list, there is a statistics summary of all existing agents' stores.

✓ Agents → Shuffled Status

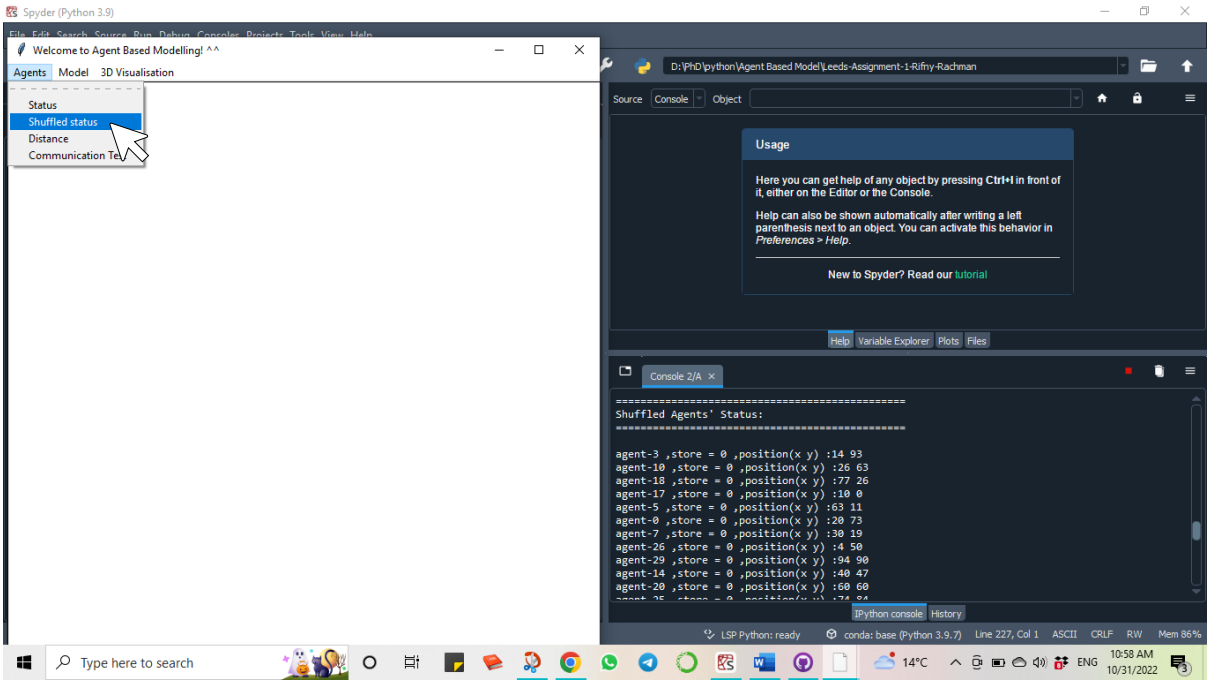

Figure 4 Agents Shuffled Status

Agents Shuffled Status feature represents all existing agents' statuses in randomized order.

✓ Agents → Distance

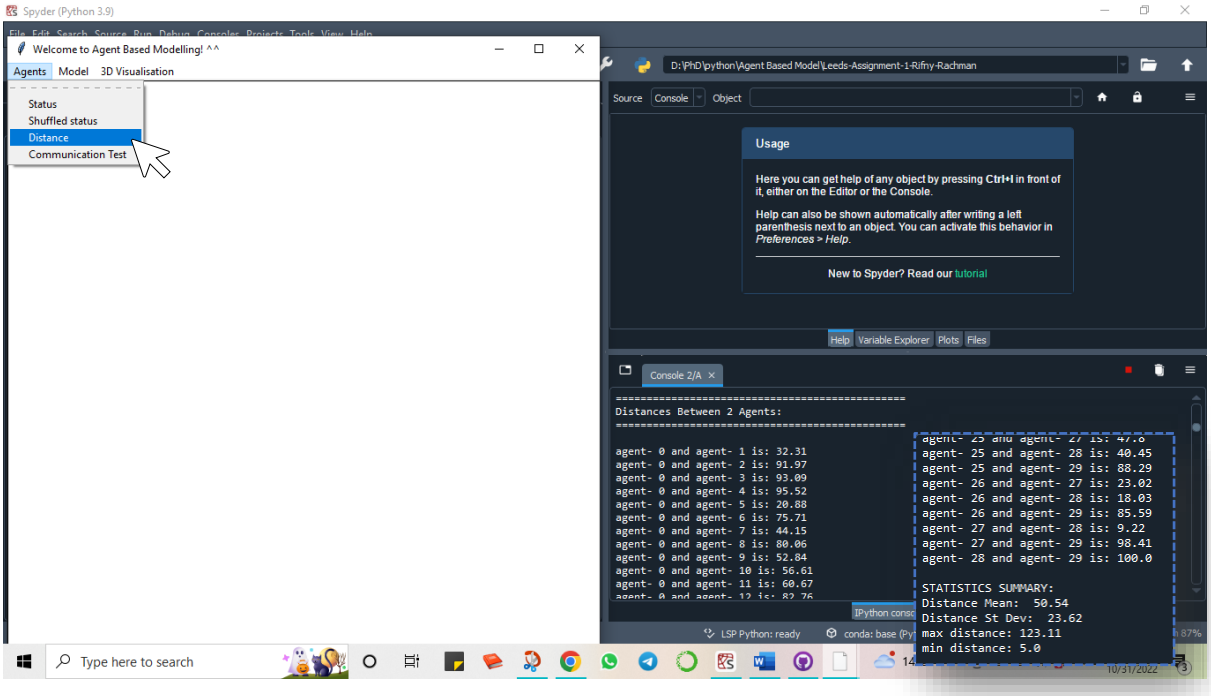

Figure 5 Agents Distance

Agents Distance feature returns distances between each couple of existing agents. At the end of the list, there is a distance statistics summary that encompasses mean, standard deviation, maximum, and minimum distance.

✓ Agents → Communication Test

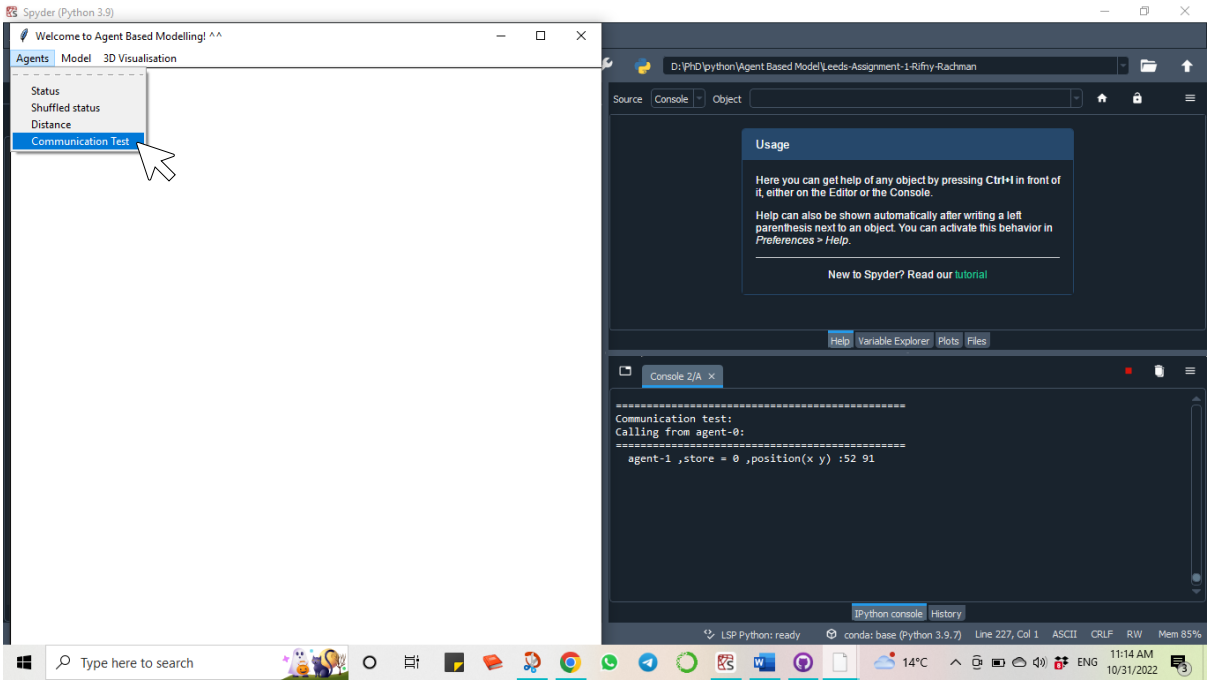

Figure 6 Agents Communication Test

With the Agents Communication Test function, users could test the interaction between agents. In this case, the program tries to call out agent-1's status from agent-0's.

✓ Model → Run

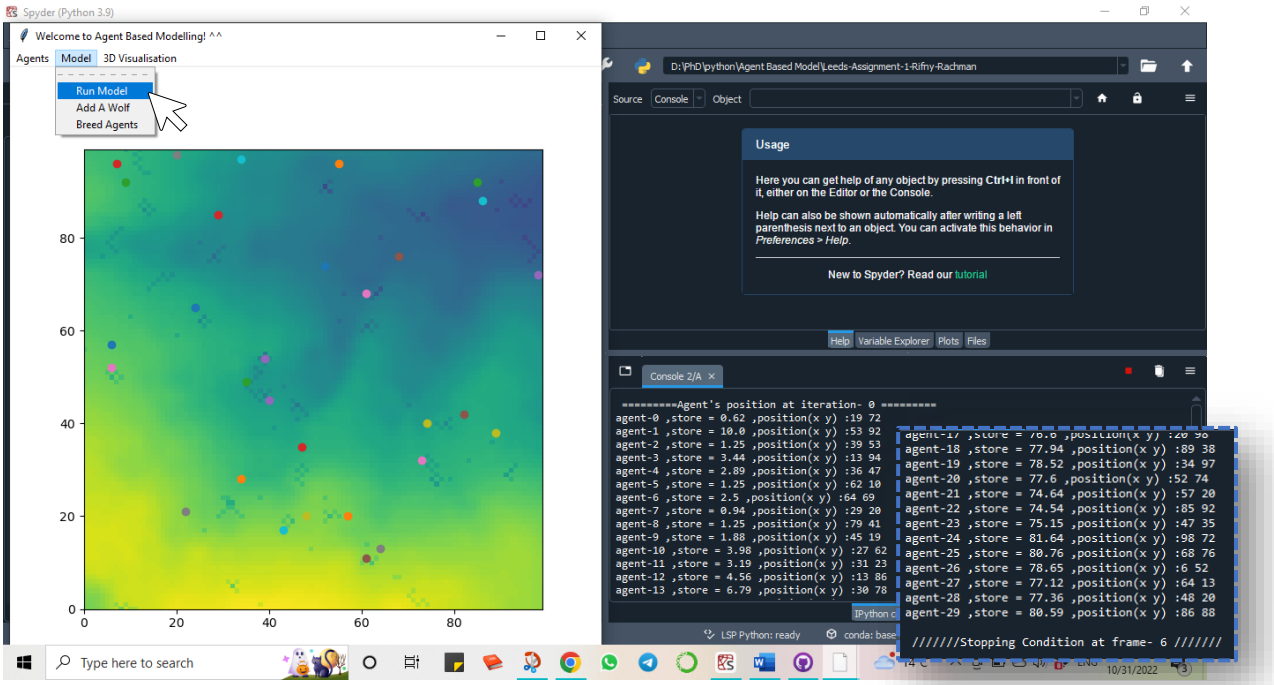

Figure 7 Model Run Model

Run Model is considered the main function of this program. Once users hit the Run Model feature, an animated graph pop-ups on the GUI window. The animation illustrates agents' behaviour, including moving, eating, and sharing with the neighbourhood. At the same

time, users could see each agent's store and position at every iteration on the console window, as well as the stopping frame.

The agents' behaviour that is animated in this function includes:

- **Move**: Agents move randomly at each iteration and move quicker if their resources reach a predefined certain amount.
- **Eat**: Agents eat from the environment and get sick if they eat more than a predefined certain value. Should the latter happen, agent's store will revert to zero.
- **Share with neighbourhood**: Agents equally share their store with neighbours within a predefined distance. However, if their resources are getting low, they will steal from their neighbourhood instead.

✓ Model → Add a Wolf

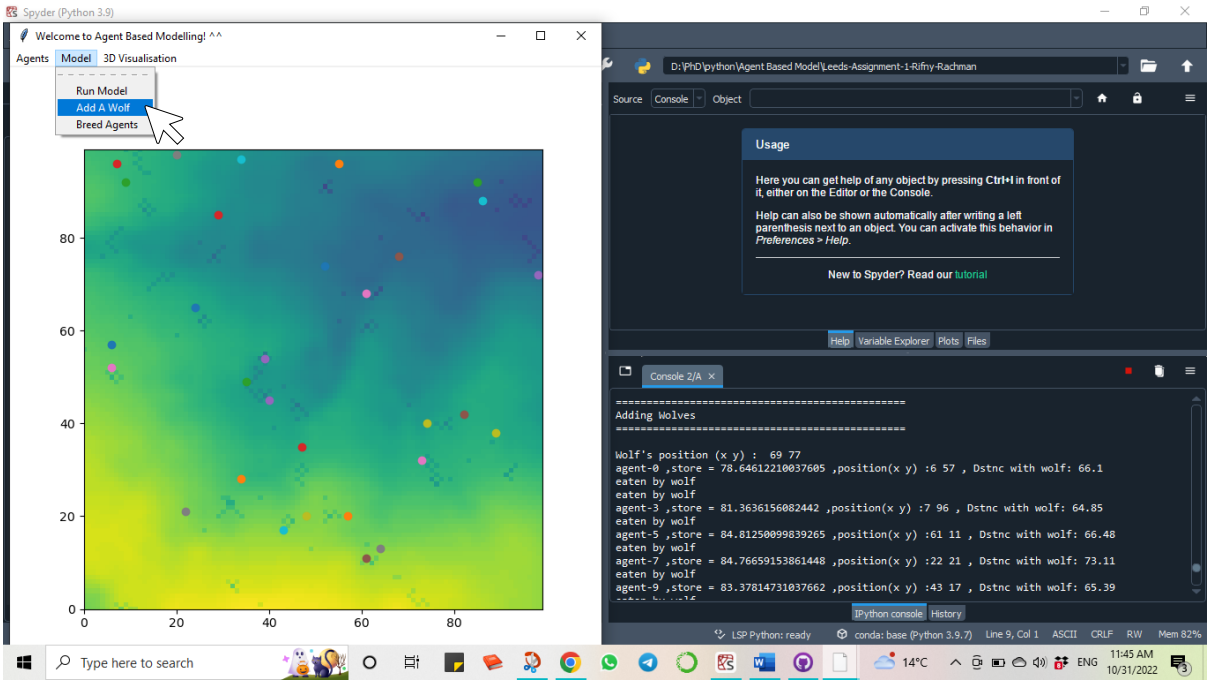

Figure 8 Model Add a Wolf

In this function, users could add a wolf as a predator who eats surrounding agents within a predefined distance. The console window portrays the wolf's position, the agents who are eaten, and the status of remaining agents as well as each distance with the wolf. If users are willing to add the wolf, they simply re-hit the button, then the second wolf would be created. The eaten agents will no longer appear on the agents' status and distance list. The following Figure 9 illustrates agents' status and distance from agents after three wolves have been added:

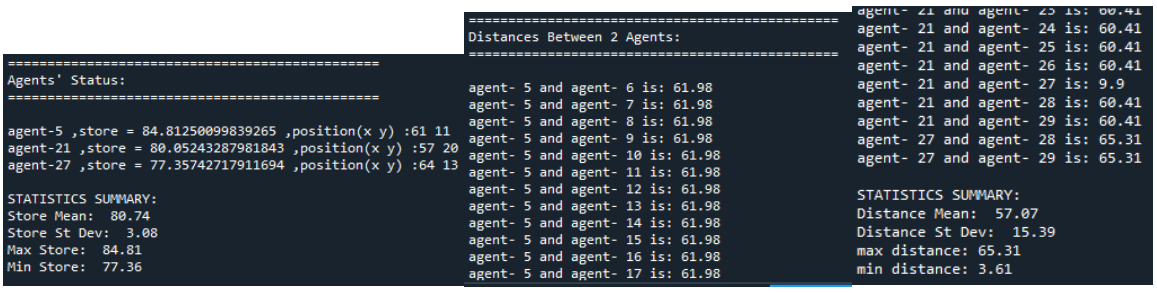

Figure 9 Agents' Status and Distance After Eaten by Wolves

✓ Breed Agents

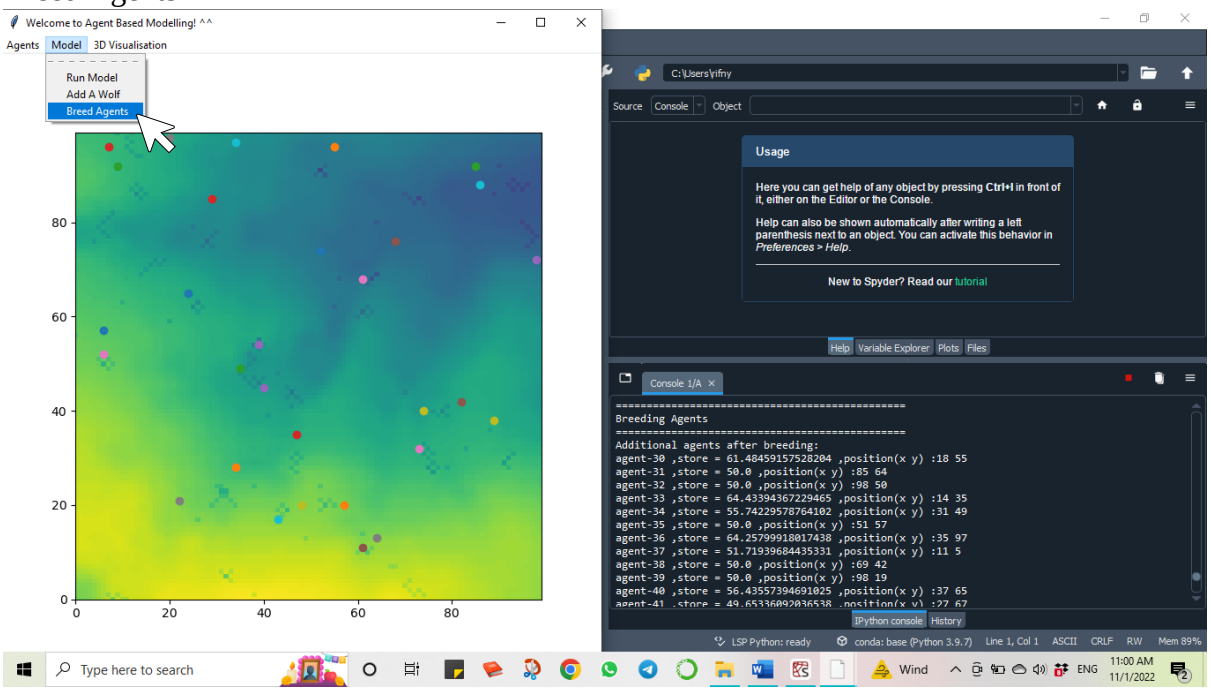

Figure 10 Breed Agents

In this function, a predefined number of agents are added to the agents' list. Furthermore, the additional agents have behaved as same as their precursors. Hence, as shown in the console window, their stores have already been filled based on a similar logic to previous agents.

✓ 3D Visualisation

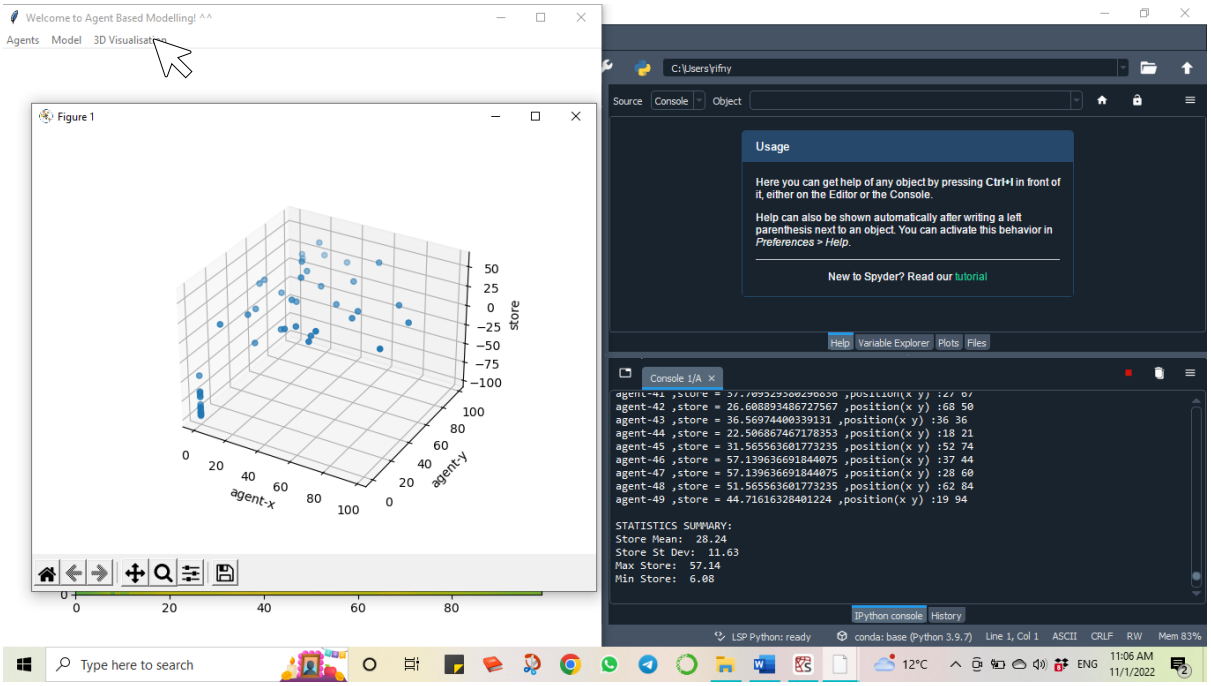

Figure 11 3D Visualisation

In this function, users could check the 3D graph representation of the latest agents' status. This 3D visualisation would be updated along with agents' status changes, for example,

after adding wolves or breeding agents. Users could also show the latest status by hitting the “Agents Status” button and referring to the statistics summary corresponding to the graph.