**READ ME**

Detailed below are the files that are attached to this project. The python script files provide detailed comments, firstly breaking down key sections into manageable chunks of code which all perform a single purpose. In addition, there are individual comments in these sections which either provide further information about what each line of code does or gives regular checks on the type of testing which was undertaken throughout.

A list of the key files uploaded are listed in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Practical session name** | **Key features** | **Files** | **Important features of scripts** |
| Agents! | Sets out object orientated ABM with agents’ properties and behaviours. | model\_agents.py  agentframework\_agents.py |  |
| I/O | Imports data as agents’ environment and gets them to interact with it. | model\_io.py  agentframework\_io.py  in.text.txt | Output to txt file |
| Communicating | Agents interact with each other by altering each other’s variables. | model\_communication.py  agentframework\_communication.py  in.text.txt | Uses time.time to investigate how long different iterations take to run. |
| Animation/Behaviour | Using Matplotlib to animate agents’ behaviour in environment. | model\_animation.py  agentframework\_animation.py  in.text.txt | Animation pop out |
| GUI/ Web scraping | Giving model Graphical User Interface and loading data to model. | in.text.txt  model\_gui.py  agentframework\_gui.py |  |
| Model development (Hunting) | Enhances Agent model by utilising two child classes (wolves and sheep). | model\_hunting.py  agentframework\_hunting.py  in.text.txt | Parent/child class example. |

**What the software is?**

This software develops over the six exercises, an agent-based model (ABM). These are used for stimulating the actions of autonomous agents (this could be individuals or organisations) with a view to assess their effects on the system as a whole. ABMs are used extensively across a wide range of disciplines, particularly wildlife ecology and management. Here we have an Agent class moving round a bounded environment, and having ever more complicated interactions with other agents. Over the practical sessions, external data is added, and the Agents are animated and finally displayed on a Graphical User Interface.

**How it can be run?**

The software is run via Python. The code is executed either in sections to document the nature of the tasks being completed at each stage, or else using the ‘run’ function in the menu. All input files are included in the project, so should be downloaded to a single location on a computer. Only the ‘model’ files need to be run.

**What is to be expected when its run?**

The model develops its complexity over time. Each time a file is loaded, the relevant modules are imported, our agents and environment are set, and then the agents interact, and these actions are in turn visually presented. First, in a static scatter graph, and subsequently in an animation, that gets shown in a graphical user interface.

**Any known issues?**

The software runs as expected. There are a few examples of experimental code which I couldn’t implement fully (e.g. extra exercises in the input-output practical). Lastly the GUI interface is glitchy. I have had issues with my Spyder installation, the code frequently crashes. However I also wasn’t able to get the Graphical User Interface to animate the web scrapped data, so had to use the random integer data instead.

**What testing was done?**

Testing is a vital part of software development. Testing can and should take place throughout the software development process, from starting to write the initial code to testing the robustness of the program prior to putting the product to market. In simple terms, this could be testing that parameters that are imported or initialised as expected, to more sophisticated testing of specific functions which may involve multiple analyses. In addition, debugging is essential to order to pick up and rectify where code fails. In the examples in this project, a combination of all three occurs. Particularly, ‘print’ statements are used throughout to check what data python returns, i.e. interrogating a list or checking the values of particular functions. This was used extensively in the ABM extension, when the single ‘parent’ Agent class was replaced by two ‘children’ (Wolves and Sheep). The interaction between the ‘Agents’ – namely the wolves checking their local neighbourhood and eating close by sheep was computational difficult as a list of sheep had to be created and then these removed in turn from the sheep list (given in reverse order). Print statements are displayed throughout, and ensure that at each stage along the process, meaningful values are being returned.

Testing also can involving timing particular functions or sections of code. This was demonstrated in the Communications practical when the time.time function was used to calculate the time used to run the same bit of code, with a single input parameter being modified each time (namely the number of iterations). Although the performance differences are relatively small on a section of code this size, on more complicated projects, small efficiency savings could be particularly valuable. In addition to time tests, more sophisticated testing should also be undertaken. Unit tests are a testing technique where individual model components are tested by the developer to determine if there are any issues. For my Agent model, testing could encompass both classes and functions. Unit tests are a testing technique where individual model components are tested by the developer to determine if there are any issues. In addition, dotests are a helpful tool to ensure code is documented and tested and the code runs as expected, helpful for both collaboration across programmers, but ultimately with users also.

**Future improvements**

As the natural environment is complex, I would have liked to have further developed my agent model to encompass more agents, possibly with different parent classes (birds and animals) and more sophisticated behaviour. I would have also liked to layer another spatial dataset on to change the nature of the environment (e.g. gradient). Furthermore, the GUI interface should be more descriptive and give the user some context in terms of the animation they are seeing and provide an opportunity to save the resulting animation.