**Python Software Development Guide – Bacterial Bomb**

**Introduction**

The python software developed to model the Bacterial Bomb scenario outlined by University of Leeds has four basic components, read in the data, process the data, display the output, and write the output to file. This guide provides detail of the development process based around the outlined model components.

* A brief description of what the software can achieve from the end user perspective, based on the scenario.
* The theory behind the design including good practices and Unified Modelling Language (UML) diagram.
* The software development process in chronological order, a diary-like format.
* Any issues encountered in chronological order, and how they were rectified.

**Intention of the Model**

The model centres around the wind raster dataset and broadly follows the standard process of data analysis (Navlani et al,. 2021). Collecting data involves reading the wind raster into the model. Pre-processing the data allows the area to be returned to a blank canvas after retrieving the bomb location. The data is then processed using the created bacteria particles and methods. After the data has been modified then the results are to be displayed in a useful and user-friendly manner. This included the key data print outputs, the density display and output text file.

**Thought Process of Model Design**

The design process adhered to agile software development principles (Sacolick, I. 2022), however some agile concepts were not viable for this project such as user testing, sprints, and team collaboration. The main agile practices used were iterative development, minimum viable product (MVP), and user experience.

The project was an iterative development, and the software was built in stages with clear aims and testing of code to ensure the correct output was achieved before moving onto the next stage of development. The biggest advantage to breaking the model build into stages is it allowed easier debugging of the code. The disadvantage to working on sections at a time is it is easier to lose sight of how the stage fits into the overall picture or interacts with yet not started blocks of code.

The projects primary focus was on getting the model to MVP, so it achieves it’s objectives. A UML diagram was produced to link the model and the module and included all the methods and variables required to build the basic model. These were then linked up and correctly ordered to ensure the model ran properly.

After the MVP was successfully working then the focus was user experience to enhance the model. This included printing key data outputs, allowing user input parameters and attempting to implement animation; however this final goal was not achieved.

A picture containing table

Description automatically generated

*Fig 1. UML diagram.*

**Model Development Process**

To ensure the model had a set location in order to map out the spread of the bacteria, a raster dataset was provided called wind.raster. This raster file was read into the model using the csv.writer function, and a 2D list was created from the raster values. This list was used to create the area for the model and contains almost all pixel values of 0 and a single pixel of 255 to represent the epicentre of the bomb.

Chart, histogram

Description automatically generated

*Fig 2. Created Area from wind.raster*

Using functions from the numpy module found the bomb epicentre from the 2D list ‘area’, as this has a unique value of 255. The returned index values are used as the starting coordinates for the bacterial spread, however it is important to remember it returns in a YX order, as opposed to the traditional XY. Afterwards replaced the value with 0 to level off the area plain.

Created the module bacterial\_behaviour which is used to affect the actions of the bacteria. Created the bacteria starting positions using the bomb epicentre coordinate values discovered in the area list. Protected the starting XY values with the get() and set() methods.

Created a parameter variable in the bacterial\_bomb\_model to determine the amount of bacteria to be created, and an empty list variable to hold the bacteria. Populated the bacteria list with the starting locations from bacterial\_behaviour times by the amount of bacteria variable.

Now working to complete the methods needed for the model in the bacterial\_behaviour module. The first method was horizontal\_movement which increased or decreased the bacteria by 1 X or Y value depending on a random chance. To determine the random change created a variable with a random integer between 0 and 100. The variable integer was then measured against various if/else statements until the correct action was identified.

The second method was turbulence to determine the height of the bacteria particles. This was done using random chance in a very similar manner to horizontal\_movement. However this was an embedded if/else statement as different actions could occur depending on the height value of the bacteria.

The final method in the module add\_density increments the area value by 1 in the 2D list on the X and Y position identified, however only runs if within the set bounds.

The bacteria particles were required to be created to enact the methods on in order to update the area list. The XY values from the bomb epicentre were assigned to the bacteria starting location and the number of particles created were taken from bacteria\_count.

Now the horizontal\_movement and turbulence method required to be run on the bacteria particles. Also included in this is an if statement to only run the actions if the particle has a height above 0, to ensure they stop where they land. There is also included a maximum iteration of the initial height doubled, which may not be the best solutions but will be explained in detail in the issues section of this guide.

Only after the movement methods have completed their iterations then the add\_density method should be run for each bacteria particle. This amends the area dataset which records the final position of the bacteria.

To show the density map then a grid should be made to create a figure and the area is shown within the X and Y axis.

Chart

Description automatically generated

*Fig 3. Density Map Example*

Four print statements were created to provide additional information to the user. The first was the bomb epicentre location. The second the amount of bacteria particles out of bounds. The third the most dense bacteria value. The fourth statement the estimated spread size.

The amended area data was written into an output text file. This contains the same amount of values as the input wind.raster, however also includes the density locations and does not include the building location.

The bacteira\_count and height were set as parameter values using the argv function. These were also given a default value specified in the scenario brief if no user input was passed in.

**Issues During Development**

Whilst attempting to retrieve the 255 index position from the 2D list ‘area’, the initial attempt was made using the enumerate function, which is ideal for indexing 1D list values. However this proved too challenging for the 2D list, so the alterative workaround was to convert the list to an array and use the functions available in numpy. Fortunately, the array variables created were not required beyond the specific job mentioned, which reduced the risk of potential data compatibility issue in later code.

Possibly the most troublesome line in the model is line 80, which provides an iteration value for each bacteria to action the movement methods. If this is not included, then the bacteria particles will only move once. A while loop was attempted instead of a for loop, for example ‘while height > 0’, however any attempted while loop seemed to crash the model when run. Therefore, a for loop integer value was required, if the value was too high then it reduced the model efficiency and if the value was too low then it may not allow the height to reach 0. As the user could change the height parameter at will a fixed value was not set, and the iteration had to be based off the height values. Doubling the values seemed mostly safe as the height had either a 70% or 100% chance of falling at each iteration.

An boundary solution such as tarus or wall bounce would not be a desirable method to employ as it would return a false reading if it was required to be used. Therefore allowing a near infinite plane would be the best solution, however if any particles did leave the area boundaries then a print return was set up to count them and inform the user.

Another of the print return was for the estimated spread of the bacteria. This is provided in metres squared and would be used to guess the amount of contaminated area. The main issue of this return is the equation to calculate the spread assumes a square or rectangle, however does not take into account the shape may differ. This was only combatted by including the word estimated in the print return, so the user does not assume the answer is absolute.

Not an issue as such, more of a confession, but whilst trying to make a default value for the input parameters research was required to be carried out to get the code. The answer provided was to pass in if the stated integer value was higher then the argv positional value use the user input, else use the set default (ettanany. 2016). When implementing this solution all tests were successful and it was kept in the code. However, it is too difficult to understand the specifics of this code solution and determine how it works.

An animation was attempted on the figure and worked in the sense the bacteria particles movement could be tracked across the area. However, the updates to the height and area variable did not appear to update beyond their immediate iteration loops whilst running in the animation code, so the animation attempt was eventually abandoned.

**References**

Ettanany. 2016. How do I assign a default value if no argument is passed in + python. *Stack Overflow.* [Online] [Accessed 05 May 2022.] Available From: <https://stackoverflow.com/questions/41028491/how-do-i-assign-a-default-value-if-no-argument-is-passed-in-python>

Navlani, A., Idris, I., and Fandango, A. 2021. Section 1: Foundation for Data Analysis. *Python Data Analysis, Packt Publishing Limited, Birmingham,* pp 8 - 13*.* [Online] [Accessed 04 May 2022]. Available From: <https://app.knovel.com/web/view/khtml/show.v/rcid:kpPDAE0001/cid:kt012W35QF/viewerType:khtml/root_slug:python-data-analysis/url_slug:getting-started-with?&b-toc-cid=kpPDAE0001&b-toc-root-slug=python-data-analysis&b-toc-title=Python%20Data%20Analysis%20%283rd%20Edition%29&b-toc-url-slug=section-1-foundation&kpromoter=federation&page=9&view=collapsed&zoom=1>

Sacolick, I. 2022. What is agile methodology? Modern software development explained. *InfoWorld.* [Online] [Accessed 04 May 2022.] Available From: <https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html>