# GEOG5995 Assignment 1 Marking Criteria

Your program/code/software will be assessed against the following criteria:

● **Readability**: The ease with which your code/software can be understood by a human. Usually helped by: an abundance of clear, concise, informative comments; the use of naming conventions; a consistent and standard source code layout (often achieved via appropriate code indentation/whitespace).

● **Structure**: The degree to which the code has been organised into relevant blocks, files and other structures as appropriate.

● **Validity**: The severity and quantity of any logical or functional errors and the presence of appropriate tests and evidence of internal checks.

● **Efficiency**: The extent to which the code reduces unnecessary computation and limits memory usage.

● **Functionality**: The overall functionality and usability of the software and any Graphical User Interface and website.

The main part of this assessment regards the Python source code and the functionality of the software produced. Of importance, but lesser importance is the auxiliary documentation.

In general the Python source code is expected to be:

* well documented (with plentiful appropriate comments that explain what the code is doing (and providing evidence of testing));
* well laid out (import statements at the top of the code, comments and code lines not too long and indentation aligned); and,
* well structured - such that the code is: concise and there is not too much repetition (so the code is more maintainable), and the algorithms are reasonably efficient.

A pass or a merit class mark may be given for this assignment without you having created a web site, but a website alone is not sufficient to achieve a pass grade for the assignment. There must be a reasonable attempt to develop a Python program consisting of at least two classes one which is the main class that is run. There should be a separate class which specifically deals with agents.

The code/software not compiling or running does not *necessarily* mean the work will be classed as a fail, especially if an earlier version of the code/software does work, but there is a failure that has occurred as a result of attempting to add some extra functionality.

The final mark will be based on a combination of elements from across the marking scheme using the following rubric:

The functionality of the Python program will be evaluated by inspecting the source code and any documentation and running the software:

* For an initial pass mark (55), the program should instantiate a collection of agents that interact with the environment, and there should be both visual and textual output from running the model.
* For an initial merit mark (65), the agents should interact with each other, and one of the outputs from running the model should be an animation.
* For an initial distinction mark (75), the model is expected to have a GUI and/or some way to input or modify parameters without having to modify the code.
* For an initial high distinction mark (85), the model is expected to have another significant enhancement as suggested in the practical documentation.

In cases where there is some of what is needed for a higher grade, but not really everything (some parts are missing and or don’t work), then a mark is given some place between using the judgement of the assessor.

Initial marks are then adjusted as follows:

* Auxiliary documentation about the software is expected to include a simple readme file, a licence file and may also include user and developer documentation. The readme should either be a markdown format or simple ASCII text file. It should provide a contents (a simple list of what all the files/directories are), outline what the software is, how it can be run and what is to be expected when it is run, and should point to the licence.) Any additional document might provide more details about how to run the software, what to expect when it is run, outline any ‘known issues’, outline any testing done and may provide a suggested development roadmap.
* Within the initial grade, the mark will be modified by between -5 and +5 depending on the quality of the auxiliary documentation:
  + The mark will be increased by 5 if the readme contains all the necessary details and any additional auxiliary documentation about the software is well presented and useful.
  + The mark will be decreased by 5 if the auxiliary data is less than useful or is non-existent.
  + The mark will be changed by -4 to +4 in other cases based on the judgement of the assessor as to the quality of the auxiliary documentation.
* Source code documentation/comments are expected to be detailed and extensive and ideally should make it clear that the developer knows what the code does. Each function and each variable should be described. Evidence should be left either in comments or in auxiliary documentation to provide evidence that the code has been tested and perhaps also timed (this may unusually clutter programs if left in comments, but no marks will be deducted for having what might ordinarily seem like being too much comment). Ideally and for top marks classes and function documentation should be in place and written in a standard way listing all positional arguments and detailing what if anything is returned.
  + The mark will be changed from -5 to + 5 depending on the quality of the source code documentation/comments.
* Up to 5 marks may be added depending on the quality of the structure and efficiency of the code and considering the validity of the model/program.
* It is expected that there is an HTML web profile page of the software/software developer and an online repository of the source code. The validity of the HTML will be tested using the W3C validator: <https://validator.w3.org/nu/>
  + The mark will be changed from 0 to + 5 depending on the quality of the website. If the HTML is valid, the web page is clear and informative and there is an online repository of code, the mark will be increased by +5. If some of the HTML is invalid, but the website is nevertheless impressive the mark may still be increased by a full 5 marks.

|  |  |
| --- | --- |
| **Grade** | **Typical Criteria** |
| **High Distinction** | **Readability**: The code/software will have excellent documentation. The software source code will contain appropriate useful comments that explain what the code does. The code will be laid out clearly and consistently adhering to a style guide. The program itself may come with structured help files which may be embedded throughout the system.  **Structure**: The Python code will have a main file that is run and a separate class file for dealing with agents. It may optionally have other classes for reading and writing to files. There will be functions that undertake tasks that are repeated. The Graphical User Interface will be clear and the website will be well structured.  **Validity**: The code will show an appropriate level of testing and internal checks, both to ensure the program’s validity and to prevent fatal issues during running. There will be checks to help prevent the system exiting in error (especially as a result of user providing unusual input values or clicking buttons and pressing keys wildly).  **Efficiency**: The code will have close to optimal efficiency in terms of the time taken to run the program.  **Functionality**: The code will include the core functionality, enhanced with additional functionality. The software will be designed to produce informative error messages that advise a user if an error is encountered. The system will be designed to prevent users making mistakes and/or that mistakes are resolved with minimum of difficulty.  Overall, the code will be of a professional release standard, or close to it. |
| **Distinction** | **Readability**: The code/software will have excellent documentation. The software source code will contain appropriate useful comments that explain what the code does. The code will be laid out clearly and consistently adhering to a style guide.  **Structure**: As for **High Distinction**.  **Validity**: The code will show an appropriate level of testing and internal checks, both to ensure the program’s validity and to prevent fatal issues during running.  **Efficiency**: As for **High Distinction**.  **Functionality**: The code will include the core functionality. The software will be designed to produce informative error messages that advise a user if an error is encountered.  Overall, the code will be of a near professional release standard with only minor usability issues. |
| **Merit** | **Readability**: The software source code will contain appropriate useful comments that explain what the code does. The source code will be laid out with only minor inconsistencies. The program/software will have some user and developer documentation that helps users understand what the program does and how to run it, and that details the thought processes behind the code functionality and development.  **Structure**: Where appropriate, the code will show some thought into the development of structural units (like classes) resulting in clear code, but perhaps not really facilitating code reuse. Any Graphical User Interface and website will show thought in their structuring, with appropriate separation of functionality.  **Validity**: The code will include standard elements of checking and elements to prevent errors.  **Efficiency**: The code may be inefficient with no evidence that work has been done to make the code efficient.  **Functionality**: The code will include most of the core functionality, with only minor issues. |
| **Pass** | **Readability**: The code/software will have limited documentation. There may be little evidence that thought has gone into code structuring or standards. There may only be minimal inline comments. There will be an attempt to lay out the code neatly, but there may be some inconsistency. Ancillary documentation may lack detail, may be unclear in parts and may be of only limited use.  **Structure**: The Python source code may only follow broad structures associated with the coding language. Where there is an expectation of code separation, much of the code may nevertheless be in a single file or block. The website may not function well as a consequence of poor structure.  **Validity**: Some errors may be encountered at runtime without this being documented.  **Efficiency**: The code may be inefficient.  **Functionality**: There will be evidence of an attempt to include the core functionality, although numerous issues may have been encountered. |
| **Fail** | **Readability**: The code/software will have little or no documentation. Inline comments will be minimal or non-existent, or may contain major errors or misunderstandings. The code layout may be confusing and difficult to read. Any ancillary documentation may be of very limited or no use. There may be little or no attempt to articulate issues encountered in developing the code/software.  **Structure**: The code will be poorly structured and may, for example, be in a single block.  **Validity**: Many errors may be encountered at runtime without this being documented.  **Efficiency**: The code may be very inefficient.  **Functionality**: There may be little evidence of an attempt to include the core functionality. |

# Feedback Sheet

In general, your overall mark will reflect the average of the categories below, however please note that the grades for the individual components are nevertheless indicative only and the overall mark may vary from this. For example, a lecturer may decide that a very nice but minor piece of code warrants a higher mark in total, despite not making a huge difference to any one of the individual categories. Please read the detailed feedback in order to understand your mark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | High Distinction | Distinction | Merit | Pass | Fail |
| Readability |  |  |  |  |  |
| Structure |  |  |  |  |  |
| Validity |  |  |  |  |  |
| Efficiency |  |  |  |  |  |
| Functionality |  |  |  |  |  |
| Overall |  |  |  |  |  |

Feedback

Areas to especially prioritise

|  |  |
| --- | --- |
| Marker | Provisional mark |
|  |  |

*Any mark given here is provisional and subject to moderation by the School’s Board of Examiners. This is to ensure comparable marking standards for all students. In a minority of individual cases moderation can lead to either the raising or lowering of the provisional marks.*