# **Assignment 2**

## Brief

This assignment is to designing and developing a program to do one of two things:

* Site suitability
* DEM processing

Details of these are provided below.

You can reuse code submitted for Assignment 1, but this will not count for much if anything in the assessment.

The submission should include: all the code and data; a [README](https://en.wikipedia.org/wiki/README); a short document of a maximum of 2000 words that explains to a user how to run the software and what to expect when it runs (optionally these details can all be included in the README); a short document of a maximum of 2000 words that details any testing done and any major issues encountered during development and how these were overcome (or not).

Submit a zip file containing all the relevant components to Minerva.

It is suggested that you set up a private GitHub repository for organising and developing your submission and download a zip file from GitHub to be uploaded to Minerva as your submission.

Source code documentation/comments are expected to be detailed and extensive and ideally should make it clear what the code does. Each class, function and variable should be described. It is expected that there will be docstrings for functions - listing positional arguments and detailing any returned values.

The README should either be a [Markdown](https://en.wikipedia.org/wiki/Markdown) file or a simple [Text](https://en.wikipedia.org/wiki/Text_file) file that. It should provide a contents (a simple list of what all the files are in the distribution) or link to a [Manifest](https://en.wikipedia.org/wiki/Manifest_file) file which provides the content details. It should declare the [Software Licence](https://en.wikipedia.org/wiki/Software_license). It should outline what the software is, how it can be run, and what is to be expected when it is run. Details of this can be provided in another document.

The need for documentation in addition to the README will depend on how detailed the README is. Evidence that the code has been tested as it has been developed is important. Some of this evidence might be in the source code files, but some details about testing might be included in the README or in a document about the development process which is to detail any remaining issues and outline any major issues that were encountered during development and how these were dealt with or remain unresolved.

The final mark is based on a combination of elements from across the marking scheme.

# Site suitability

A company that produces rock aggregate wants to explore three factors they consider important in locating a factory in the UK. They have asked you to produce some software that helps them do this. They have provided two dimensional raster data for each factor with values in the range [0, 255]. The higher the value of a factor, the more suitable the location is for the factory. Each factor is to be multiplied by a weight and the weighted factors are to be added up to give an overall suitability for each raster location.

The company wants the software to allow them to easily choose the factor weights and visualise the suitability.

Write some software which does the following.

1. Reads the raster data and displays them.
2. Multiplies each raster by a factor, adds the weighted rasters together and rescales the resulting raster to have values in the range [0, 255].
3. Displays the result raster and writes this to a file.
4. Provides a GUI that allows the user to choose the weights by means of ‘sliders’ and that displays the result.

Hints:

* Use lists (of lists) to store the raster data.
* Process the data row by row and column by column and create a combined weighted and summed value raster, then rescale this raster.

Files:

Raster data text files written in row major order (top to bottom and left to right). Each line is a row of the raster:

* [Geology](https://agdturner.github.io/resources/assignments/geology.txt)
* [Transport](https://agdturner.github.io/resources/assignments/tranpsort.txt)
* [Population](https://agdturner.github.io/resources/assignments/population.txt)

# DEM processing

An extreme sports holiday company wants you to find some extreme gradients where they might potentially set up some sporting activities. You are given a 2D raster data set of elevations above sea level.

Write a program which does the following.

1. Reads the raster data and displays it.
2. Calculates the maximum slope for each row and column (pixel) of the raster by considering all the slopes between the pixel and the 8 nearest pixels. (Think about what to do at the map edges.)
3. Saves the maximum slopes to a file in the same format as the provided raster data.
4. Provides a GUI that allows the user to choose the raster data file to load and that displays the result and on top of this the most extreme locations in terms of maximum slope.

Files:

* [DEM text file](https://agdturner.github.io/resources/assignments/slope.txt) - values are written in row major order (top to bottom and left to right). Each line is a row of the raster.

# **Marking Criteria**

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

● **Readability**: The ease with which your code/software can be understood. Usually helped by: an abundance of clear, concise, informative source code comments; the use of naming conventions; a consistent and standard source code layout (achieved via standard and consistent code indentation, blank lines and the use of standardised documentation syntax); and producing appropriate ancillary documentation as part of the submission.

● **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 internal checks. (Evidence of testing can be provided in documentation as well as in source code.)

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

● **Functionality**: The overall functionality and usability of the software.

| **Grade** | **Criteria** |
| --- | --- |
| **High Distinction** | **Readability**: The software will have excellent documentation. The source code will contain comments that explain what the code does and will be neatly organised into code blocks that start with a comment. There will be docstrings for functions as appropriate and these will detail any parameters and returned objects. Code will be laid out clearly and consistently. Ancillary documentation will be of excellent quality and will detail the development process and how to use the software.  **Structure**: The code will be appropriately divided into structural units (modules, classes and functions) that enhance the reusability and readability of code.  **Validity**: There will be clear evidence of testing that ensures the code does what is intended. There will be checks on inputs to ensure user provided parameters are correct and that data used is in the format expected. The results will be as expected.  **Efficiency**: The code will be efficient in terms of both memory and processing times. Parts of the code will have been timed and optimised.  **Functionality**: The code will include all the functionality for the task chosen which will work flawlessly.  Overall, the code/software will be user and developer friendly and be of a professional release standard. |
| **Distinction** | **Readability**: The code/software will have good documentation. The source code will contain comments that explain what the code does and will be neatly organised into code blocks that start with a comment. There will be docstrings for functions as appropriate and these will detail any parameters and returned objects. Code will be laid out clearly and consistently. Ancillary documentation will be of good quality and will detail the development process and how to use the software.  **Structure**: The code will be appropriately divided into structural units (modules, classes and functions) that enhance the reusability and readability of code.  **Validity**: There will be clear evidence of testing and the results from normal use will be as expected.  **Efficiency**: The code will be efficient in terms of both memory and processing times. Parts of the code will have been timed.  **Functionality**: The code will include all the functionality for the task chosen.  Overall, the code/software will be user and developer friendly and be close to a professional release standard. |
| **Merit** | **Readability**: The code/software will have good documentation. The software source code will contain comments that explain what the code does although these may be improved. There may be some minor inconsistencies in code layout. Ancillary documentation will describe development process and how to use the software, but this could be significantly better.  **Structure**: The code will be structured to avoid large amounts of repetition but this might be improved in some ways.  **Validity**: The evidence of testing could be improved. Results from normal use will be as expected although there might be minor issues.  **Efficiency**: The code may have some inefficiencies, but these do not severely affect processing times.  **Functionality**: There will be an attempt to implement all the functionality for the task chosen, but there will be issues with some of this. |
| **Pass** | **Readability**: The code/software will have documentation, but this could be majorly improved. The software source code will contain some comments that explain what the code does, but some of this may be unclear and it would be better if there was more of it. Ancillary documentation will lack some important detail and contain minor errors, but not many.  **Structure**: The code structure may be majorly improved. There may be large amounts of unnecessary code repetition.  **Validity**: There will be little or no evidence of testing. Results from normal use will be as expected although there might be minor issues.  **Efficiency**: The code will be inefficient in parts making run times unnecessarily long.  **Functionality**: There will be an attempt to implement all the functionality for the task chosen, but there will be several minor issues or some major issues with some of this. Nevertheless, this will be a satisfactory attempt. |
| **Fail** | **Readability**: The code/software will have little or no documentation, or what there is may be unclear or misleading.  **Structure**: The code is poorly structured.  **Validity**: The results are not as expected and evidence of testing is insufficient.  **Efficiency**: The code is very inefficient and there has been no attempt to address this.  **Functionality**: There is an insufficient attempt to develop the functionality for the task. |

# 

# **Feedback Sheet**

In general, your overall mark will reflect the average of the categories below, however the overall mark may vary from this and is based on the overall judgement of the assessor. Please read the detailed feedback in order to understand your mark.

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

Feedback:

| 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.*