|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Excellent  (>70%) | Good  (60-70%) | Basic understanding  (50-60%) | Poor  (40-50%) | Fail  (<40%) |
| Part 1 - Encryption and decryption  14 marks | Functions imported from another file and/or as class methods [+1] | Fully working implementation of  encryption and decryption  [+1]  Pass message and rotation as function arguments  [+1] | **Implementation of encryption and decryption.**  **All inputs given by user (none defined in program)**  **Use of functions**  **Program prints the output**  **[7]**  Program accepts messages with multiple lines.  [+1]  If inputs missing/incorrect, user prompted.  [+1]  Numbers punctuation spaces unchanged.  [+1]  Returned message is upper case only  [+1] | Poor implementation of encryption and decryption [-1]  User not prompted for some/all inputs [-1]  No use of functions  [-1] | No significant implementation of encryption and decryption  [-2]  No inputs to program  [-2] |
| Part 2 - Analysing messages  8 marks | Function to compute metrics  [+1]  Functions imported from another file  [+1] | 1-3 metrics computed correctly  [+1]  4-5 metrics computed correctly  [+2] | **Outputs a value for each of the required metrics:**   1. **Number of words** 2. **Number of unique words** 3. **(Up to) The ten most common words sorted by their frequency in the format** 4. **Minimum, maximum, and average word length** 5. **Most common letter**   **Metrics collected on unencrypted data only**  **Metrics printed after encryption/decryption**  **[3]** | Metrics collected on encrypted data  [-1]  Metrics printed before encryption/decryption complete  [-1] | Missing metrics  1-3 metrics missing  [-1]  4-5 metrics missing  [-2] |
| Part 3 - Messages from a file  8 marks | Functions imported from another file  [+1]  Functions implemented as class methods  [+1] | Properly working file reading  [+1]  File closed after opening  [+1] | **Option to read from file added**  **If filename invalid, user prompted to try again**  **Program then continues to work as before**  **[4]** | User not given choice of f  **[-1]**  No prompt to try again if filename invalid  [-1] | Option to read from file not added  [-1]  Program crashes if filename invalid  [-1] |
| Part 4 - Automated decryption  14 marks | Use of decrypt function within auto-decrypt function  [+1] | If matches discovered present the line to the reader  [+1]  Ask if the line has  been successfully decrypted  [+1]  If no, continue to increment until first line successfully decrypted  [+1]  If yes, decrypt rest of file  [+1]  Fully working auto-decryption  [+1] | **Auto-decryption option added**  **Import words.txt**  **Check decrypted message against words.txt**  **[7]**  First line checked before decrypting whole file  [+1] | Use of words.txt not imported  [-2]  Use of words.txt not used auto-decrypt not implemented  [-2] | No significant attempt to implement auto-decryption  [-3] |
| Part 5 - Using Data From imported les  14 marks | Correct bending stiffness value and  units N/mm [+1]  Correct format of output .csv file (i.e. bending stiffness as new column)  [+1] | Correctly computer RMSE  Label RMSE on plot  Trend line fitting correctly implemented  [+1]  Plot features (e.g. axis labels and legend)  [+1]  Bending stiffness computed for each beam  [+1] | **Run decryption function on douglas\_data\_encrypted.txt**  **Plot the knot ratio against the bending strength of the beams.**  **Attempt to compute RMSE**  **Attempt to compute bending stiffness**  **Plot saved as .pdf**  **[7]**  Trend line on plot  [+1]  Data set saved as .csv file  [+1] | Decryption function not run on douglas\_data\_encrypted.txt  [-1]  Plots missing  [-1]  Knot ratio plotted against bending stiffness incorrectly  [-1]  Plot not saved as pdf [-1]  Filename defined as programme variable rather than user input  [-1] | No Attempt to compute RMSE  [-1]  No Attempt to compute bending stiffness  [-1] |
| Part 6 - (\* Optional) Enhancements  12 marks | Implementation of 2 or more extra features  [+12] | Implementation of 1 extra feature  [+6] | N/A | N/A | N/A |
| General implementation  10 | Robust implementation, e.g, catches wrong input from users, allows negative rotations  [+1]  Use of multiple files and imported functions  [+1]  Use of classes  [+1] | Elegent implementation  [+1]  Efficient code (Avoids repeating/redundant code as much as possible, e.g., statistics calculated right at the end, once, or moved to functions for reuse)  [+1] | **Sensible variable names (consistent, use of underscores etc)**  **Indicating Parts and subsections of assignment with comments**  **Useful and sensible comments in code**  **[5]** | Poorly chosen variable names  [-1]  Poor use of comments  [-1]  Not indicating parts and subsections  [-2] | No comments  [-1] |
| Report  20 marks |  | Analysis – what worked, what could have changed  [+ 5 – 10 depending on quality of analysis] | **Provide an example of their design decision**  **Reference any external code or ideas used (e.g. use of ASCII functions (ord and char) requires explanation)**  **[10]** | Significantly over/under the page limit (< half a page, > 2 pages”)  [-2]  Extensive waffle e.g.: “I really enjoyed writing the code…”, “I worked a lot on the code….”  [-2]  Not eferencing external code or ideas used  [-2] | Plagiarism  [-4] |

It is crucial to give useful feedback to the students. This feedback should help them significantly to write excellent code for their second submission. Here are couple of points that you should consider:

* Start with positives about the project if you can, such as the progress they made
* Be constructive in your criticism, e.g., the code could have been written more tautly by using a function for reading data from a file
  + This feedback should be mostly based on what is taught; limit your expectations a little, but provide additional suggestions if you think they would help
* Comment on overall structure
* Is the code readable? What could be improved
* Where did they score extra marks, or lose marks and why?