# Assignment 1
## Code peer-review

**Peer review results**

Creator of the code: Anonymous
Code reviewer (optional): anonymous

| | |
|---|---|
| **Topic 1: Code behavior and correctness.** The aim of this section is to evaluate whether the code is running as expected and whether its output is correct.<br><br>To assess this code for correctness, use the provided assignment files (sample_set1.txt and sample_set2.txt). Use these files to answer the questions below. | |
| Were you able to run the code with the requested command?<br><br>`from main import similarity`<br><br>`similarity(set_1='set1.txt', set_2='set2.txt', outfile='similarity.txt')` | □ Yes<br>□ No |
| If you answered no to the previous question, could you find a workaround? Comment on the workaround (if any) and whether it was difficult to find. | □ Yes<br>□ No<br><br>Comments on the workaround: I had to change the filenames to sample_set1.txt and sample_set2.txt. He called the function in the main.py file already with those names. |
| The output of the computed similarity function should be 0.59.<br>Did you obtain the correct value? | □ Yes<br>□ No<br><br>In case of no, write the value you obtained in your calculation: 0.52 |
| (Optional) If you answered "no" to the previous question, were you able to spot the mistake in the code? If so, help your peer find it, by providing indications. | I believe he only compares the last index of an interval of list 1 with the first index of the interval in list 2. While he should compare both indexes of the intervals with each other. The overlap count is therefore too small |
| **Topic 2: Code clarity.** Code clarity can be judged by assessing how easily the code can be understood and interpreted by other developers. Clear code should have clear and meaningful variable and function names, concise and well-organized code structure, appropriate comments, and adherence to coding conventions and best practices.<br>(Score each aspect from 0 to 10, with 10 being the best) | |

| | |
|---|---|
| How much does the code follow aspects of good programming style (e.g., PEP8 style)? | □ Greatly! Good job. 😀<br>□ Most times. 😊<br>□ Almost never. 😕<br>□ Never. 😫 |
| You can comment on specific aspects of style, like the ones below. If you're unsure about the score, have a look at PEP8 style guideline (as an example of style and usage). | |
| • Correct indentation. | Score 10 |
| • Naming conventions (clarity, consistency, etc.) | Score 6 |
| • Reasonable usage of line comments. | Score 8 |
| Use this space to provide comments on the scores assigned above. | Avoid using variable names called 'a' or 'b'. The usage of comments for me is a little excessive, but they are clear and most of the time meaningfull. |
| As you know, code documentation is essential as it provides valuable insights into the purpose, functionality, and usage of code, making it easier for other developers to understand, maintain, and collaborate on the codebase. In this section, you will have the opportunity to help your peers improve their documentation. Have a look at this link if you're struggling with your review. | |
| • *Function and Class Documentation*: clear and concise documentation, including a description of their purpose, input parameters, return values, and any exceptions they may throw. | Score 10 |
| • *Variable and Constant Documentation*: Document the purpose and usage of variables and constants, especially if their names alone are not self-explanatory. | Score 8 |
| • *Code Comments*: Include comments to explain complex logic, important decisions, or any non-obvious code segments. Comments should focus on the why rather than the how, as the latter should ideally be expressed through clear code. | Score 8 |
| Use this space to provide comments on the scores assigned above. | Variables are well explained and clear in what they are used for. Especially doc |

| | |
|---|---|
| | strings are very elaborate. |
| **Topic 3: Code structure.** Code structure significantly impacts the readability, maintainability, and scalability of software projects. A well-organized codebase enhances readability, allowing others (and you) to understand and navigate the code more easily. It promotes maintainability by enabling efficient troubleshooting and updates, while also facilitating code reuse and extensibility, empowering you and others to enhance the overall quality and efficiency of the software.<br>(Score each aspect from 0 to 10, with 10 being the best) | |
| • *Modularity*: Assess how well the code is organized into logical modules or functions, with clear responsibilities and separation of concerns. | Score 8 |
| • *Code Duplication*: Identify any redundant or duplicated code blocks. | Score 8 |
| • *Code Length and Complexity*: Assess the length and complexity of functions or methods. Long and complex code blocks can be harder to understand and maintain. | Score 8 |
| Use this space to provide comments on the scores assigned above. | The final calculation of the similiarity is done in the __init__ function, which I wouldn't do. I would put this into a calc_score function or so. |
| (Although not mandatory,) was object-oriented programming used? Was this useful to reduce some of the complexity? | Write your comments here. |
| **Topic 4: Reflections.** Use this opportunity to reflect on the code you have seen and compare it with your own code. Provide your comments below. | |
| How different was the adopted solution from yours? Please provide a comment. | In this case a class was used in order to calculate the similarity score. Furthermore the subfunctions are pretty similar in meaning |
| What are your main recommendations to improve this code? | I would reduce the amount of comments a little and improve the functionality of the code. |
| Is there something that you particularly liked about the code solution? If so, use the occasion to inform your peer about it! | The fact that you started on a class is a sign of thinking a step ahead, for example when using this as a loss function in a deep learning model. So yes I like that! |

| | |
|---|---|
| (Optional) Use this space to write additional comments you might have on the code that we forgot to ask. | Write your comments here. |

Document creator: Francesca Grisoni, f.grisoni@tue.nl. Reach out via Slack in case of questions.