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. 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?
 	
`from main import similarity`
 	
`similarity(set_1='set1.txt', set_2='set2.txt', outfile='similarity.txt')`
	◻ Yes
	[✓ ◻ 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] ◻ No He called the function in the main.py file already with set names. Therefore when importing in another file always the called function in main.py will also be called
The output of the computed similarity function should be 0.59. Did you obtain the correct value?	[✓ ◻ Yes] ◻ No
(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.	-
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. (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. 😃 [✓ ◻ Most times. 😊] ◻ Almost never. ☹ ◻ 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 8
● Reasonable usage of line comments.	Score 10
Use this space to provide comments on the scores assigned above.	The code is very readable and to the point. However, I wouldn't advise to use dutch words for variable names. Furthermore, I wouldn't change a thing.
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. (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 7
● *Code Duplication*: Identify any redundant or duplicated code blocks.	Score 10
● *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 10
Use this space to provide comments on the scores assigned above.	The code is very concise and clear. Little lines of code are used to achieve the goal, very efficient. However, the functions are all put inside the function similarity. This could be choice, however if I would want to just read a set it wouldn't be possible. I would advise to put the functions outisde the similarity function to make them accessible.
(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.	All subfunctions are put inside the main similarity function. This could be a design choice, but I wouldnt make it in this case.
What are your main recommendations to improve this code?	Not make use of dutch variable names
Is there something that you particularly liked about the code solution? If so, use the occasion to inform your peer about it!	The lines of code needed is very little, which makes the code very readable and to the point. Therefore it is very comprehensive.
(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.