I noticed differences between the object-oriented approach in assignment one and the functional programming approach in assignment two. In the OO approach, I used classes like FileStructureChecker and DocStringChecker that maintained state through instance variables like self.filepath and self.tree. They had methods to check for the variables, like the file structure and docstrings.

In the FP approach, my thought processes had to shift to breaking down the problems into tasks instead of focusing on entities like file structures and docstrings. In the new design, the functions didn’t rely on maintaining state. Information was passed between functions in a more modular style, like how the generate\_report function passes filepath and tree between analysis functions - for example, find\_file returns the filepath that is passed to analyze\_file\_structure. This reduced the use of mutable variables and the logic was in smaller functions instead of classes, like analyze\_file\_structure and generate\_docstring\_report, instead of the DocStringChecker and other checks being in separate classes in the OO design. This means the logic was centralized for every aspect of the style report.

My thought process was different going into it because I had to rethink the design and it felt less intuitive at first, but in the end, it was clear that the code was easier to maintain and change if necessary. It reduced the complexity and allowed for more flexibility. Both the FP and OO design have advantages and disadvantages, but I overall like the FP design more.