Assignment 2 Comparison

When refactoring my code from an object oriented approach to functional programming, I had to consider mostly how the code was contained within itself. Because the object oriented requirements had us code only two classes, the style checker and the file converter class, it forced me to organize my code structure and process in a two-step sense. On the other hand, in functional programming, because of the many different functions present in the program, I was considering the structure of the code on a more step by step basis, thinking about where each function would be necessary in terms of placement and chronology. Also, I had to focus much more on the fact that functional programming does not allow mutable variables, which forced me to simplify and break down the structure of how I thought about my code. For example, rather than changing shared state or using global variables, I made sure to pass everything I needed clearly to the function as arguments. Additionally, I did not change data, used constructs that returned new collections. The functions that could be considered “not pure,” like reading and writing the file, I wanted that to be “removed” or differentiated from the other pure functions in the system. For the object oriented programming, I tried finding things that could be state or variable and created ways to manipulate that state, like methods that modified the inner details or states of the objects. The function-oriented approach allowed for a more modular and structured design. However, OOP makes it easier to encapsulate data and structure code intuitively, whereas the focus on pure functions and immutability in FP results in code that is more predictable and more parallelizable. Overall, I think I prefer the functional programming approach, mostly because I think the process of needing to simplify the functions in order for them to remain pure allows me to understand the code better and know the next step I need to take when programming.