This document will serve as a brief writeup to shed light on my design process and how I went about doing what I did and why.

The first thing to discuss is the functionality of the program, that is, what does it do? This program allows a user to organize the files in a selected directory tree. They can sort them by their file extensions (.doc, .pdf, etc.), by their filename and by their filename into alphabetized folders. I did not allow the user to alphabetize the extension folders after some feedback that it made it a little tedious and unnecessary as there is not as much variability in extensions as there is in filenames.

The second function this program allows for is the batch renaming of all files ending in one extension to another extensions. In other words, the user can change all “.zip” files in one directory into files ending in “.rar”. I constrained this to just one directory level instead of the whole tree to avoid catastrophic user errors. As in, the use cannot accidentally rename important files in system directories.

The last function this program allows for is the deletion of all empty directories in a directory tree. This mainly serves as a cleanup for the sorting function of the program. I chose to make it an explicit function to avoid deleting directories the user might want to keep.

In general, I took precautions to avoid tampering with important system files on accident. The sorting functions skip over hidden folders and files. This also adds an extra layer of functionality in where the user can store files in a directory that begins with “.” If they want to exclude those files from the sorting process.

I broke the program into three main files. My reasoning behind this was abstraction and readability. I tried to limit what a given function did to limit possible bugs and keep to the idea that a function should not be too complex to allow for easier debugging. I have the main file where the program is instantiated. Every function in the main file calls a function in the options file. Likewise, all functions in the options file calls to only functions in the modules file. I chose this hierarchal design to avoid complex bugs. If something breaks, then I just follow the string to the root cause.

The greatest difficulty I encountered was converting the CMD user interface into an actual GUI with Tkinter. I am sure there are other libraries that may have been easier, but Tkinter had a lot of documentation and tutorials allowing me to pick it up and really delve into it. It was difficult not because I had to learn a new technology, as far as technologies go, Tkinter is intuitive, but because I had to refactor my code to work in a different way. This resulted in me scrapping functions as they were replaced with Tkinter functions or just not needed anymore. Thankfully, the hierarchical approach I took in my design made it a little easier as I only had to tweak my code instead of completely re-writing it. The only new code I wrote was the main file. All that code is from scratch.

All in all I had a lot fun with this project. It was a great experience and really let me run wild. I learned a great many new things and overcame some challenges, which I personally find fun. Growth is often measured by the challenges we can overcome and the possibilities we can achieve. I hope I have achieved in creating a nice little program that is easy to use and serves as a showcase of everything I have learned thus far.