As of 5/4/22 I have built the foundational elements of my tool. Given a zip file, it will parse the binary for local file headers, checking the signature as an indictor that it should continue. It then stores every field in a corresponding dictionary before moving onto the next one. Once an invalid signature is detected, the procedure defining dictionaries for the local file headers is concluded. It will then print the dictionaries. By default, python is capable of stringifying key entry pairs, this is what is currently used to display the structure. This approach has a messier display than I’d like, the keys themselves contain metadata that I later use to encode dictionaries back into binary; the methodology is that I can modify and recreate binary for different structures by having an easier method to interface with them. In a final product I will likely strip the displayed structures of metadata. The next two procedures then begin to parse for central directory headers and the end of central directory record. In a similar fashion, it stores the structure into a dictionary and concludes on an invalid signature before printing it.

My plans moving forward:

I need to create a logical pair between central and local directories. I think by doing this I can potentially mitigate against offsets that don’t make sense, but another indicator there is a problem stems from the number of entries alone. In that case, I can restore missing data and generate missing structures. I need to program the logic to make those decisions. I have functionality to replace specific headers, this helps modify values, but I need to make a way to append binary to certain areas. Modifying of the file has to be done by request, so to do this I’ll implement a flag cmd line system.