Reference file Validation tool:

1. The tool should accept one or more files as input. For example I can ask it to validate:

jwst\_nirspec\_dark\_0037.fits or

all files in a directory or

jwst\_nirspec\_\*

For now we will just concentrate on the FITS files but we also have asdf files that we might try to validate later.

1. Should check the primary header (but it could be a case where there are several extensions in the file and we want to check the information in each of them. For now we will just concentrate on the primary header.
   1. Should check that the keywords listed in the file required\_keywords.txt for that particular type of file and instrument are present in the header of the file
   2. Should check that the keywords common to all files (the list I gave you) are also present in the file
2. Should check that the required keywords have valid values (this information is in the file valid\_parameters.txt)
   1. Check that these have the required format, in particular for the useafter date but also that the others
   2. Should check that the header keyword has only a valid value for that instrument. At this point I think that we only give one value per keyword. I don’t know if at one point we can have more than one. Probably for now only one value per keyword but keep in mind this can change.
3. Should check all the file and indicate which are errors without exiting before finishing checking the file. There are keywords that can have instead of errors warnings. E.g. Author. (send me the list of keyowrds that should be in all files and will let you know which can be warnings). The important thing here is that whomever runs the tool can find all the problems in one run.
4. Should check that the format of the file is correct (this for the future but to keep in mind).
   1. For example, if the file is supposed to be an image with dimension (x,y), we should be able to check it. This is in particular important for subarrays. At one point we should have another table that provides with the dimensions of all the subarray names and we should check that the files are indeed that way. This might apply also to the number of integrations in a file (e.g. darks)
   2. There might be other cases when the files should have many extensions sometimes you know what type of extensions; eg. DQ, ERROR (will give these to you later)
5. Note that I put all the information in one file for all the instruments. I think this might be the easiest way to keep track of these things but you might propose to make one file per instrument that has both the required and valid information in one file.
6. The list of valid values as well as the list of required keywords might change. So once you put these in a file in the format you choose, share these with me (github is fine) so I can update them if needed (or at least download your version, make changes and give it to you with updates).
   1. Also, it is possible that the list of valid values have to be subdivided even further. For example, for now it just only list the valid values per instrument but might be the case that some of these values are only valid for that instrument and for a particular observing mode or detector. Make the code robust enough to be able to add those things in the future.
7. One other thing I need to give you later is the information for tables. In that case we want to access the table extension and check that all the columns for a given table are present and that in one or more columns only valid values are present. For example, the table can have a column called “filter” and we might need to check that only the valid filter values are present. The table might have more than one column to check. Make the code robust enough to add a check to as many columns as we need (maybe easier if we do check them one at a time?).