Preparing and reducing images for processing

1. Mounting google drive to the linux machine allows the user to copy files using unix commands rather than using the google drive web interface to zip and download in chunks.
    a. `google-drive-ocamlfuse ~/google_drive`
    b. `ls ~/google_drive`
        i. should show google drive directory
2. Copy files to the desktop (or where ever makes you happy)
    a. `cp -r 20180906 ~/Desktop/`
        i. -r recursively copies all nested folders and files in the main directory
3. Create raw directory and copy files (for backup because downloading takes a while)
    a. `mkdir raw`
    b. `cp *.fts raw/`
4. Create a calibration folder where you put both the flats and biases in
    a. `mkdir calibration`
    b. `cp Auto*.fts calibration/`
    c. `cp Bias*.fit calibration`
5. Create folders for each filter (must analyze them separately)
    a. `mkdir V w z`
6. Copy all science files to each filter folder
    a. `cp *-v.fts V/`
7. Rename all files to end in .fits using **nsb_prepare.py -r** option
    a. `nsb_prepare.py *.fit -r`
8. Navigate to calibration folder
    a. Create master images in calibration using **nsb_masters.py *.fits**
        i. `nsb_masters.py *.fits`
            1. This goes through and sorts biases from flats in different filters
            2. Creates a masterbias
            3. Then subtracts the masterbias from the flats
            4. Normalizes the flats
            5. Then creates masterflats
        ii. Verify the master flats and bias look alright
9. Now to prepare to reduce V data
    a. copy masterbias and masterflat-V.fits to V directory
    b. navigate to the V folder
    c. Make sure they are renamed to end in .fits
    d. `nsb_reduce.py kp*.fits -b masterbias.fits -f masterflat-v.fits`
    e. Verify the reduced images look ok
    f. If needed, you might need to mess with the rotation of the images
10. Prepare the images for PP using **nsb_prepare.py kp*.fits -c**
    a. `nsb_prepare.py kp*.fits -c`

   b. This injects the headers, calculates the moon distance, and crops the images.
      i. Crops 500 pixels off on each side.
   c. This process will also tell you which images do not have WCS info injected, that means these were not successfully solved by astrometry.net or pinpoint. Remove them before continuing.

THE IMAGES ARE PREPARED AND REDUCED FOR PHOTOMETRYPIPELINE

Running the images through PHOTOMETRYPIPELINE

PHOTOMETRYPIPELINE process is quite slow. I like to run it overnight. It does get hung up on some images (why? It happens when it does not have any stars available in the GAIA or TGAS astrometry catalog to match to the field)

1. Start the pipeline on all images using **nsb_process.sh**
   a. `nsb_process.sh kp*.fits`
      i. If it hangs, use control-c and it will exit out of the current process. It will continue onto the next part. You do not have to exit and restart.
   b. Again, this takes a long time. Speed up the process by cropping to a smaller frame or install the linear algebra package with threading enabled.
   c. You can change catalogs and more in mytelescopes.py in PP folder
   d. I have modified PP with the correct transformations and a backdoor to get the night sky brightness information we need.

DO NOT UPDATE PHOTOMETRYPIPELINE OR YOU WILL NEED TO REWRITE ALL OF MY WORK

A backup is made on the google drive in Sky Brightness.

Night Sky Brightness Calculation

1. Now you will have a nsb_FILENAME.txt for each file.
   a. This contains the zeropoint and all the goodies for NSB
2. Use **nsb_analyze.py** to calculate the zeropoint
   a. `nsb_analyze.py nsb_kp*.txt`
   b. Creates **nsb_data.csv**

You now have the main file with the night sky brightness measurement. The remaining steps are to plot the data, match SQM data with each file, match boltwood data with each file, package it all together for LL, and create a readme with information about the package for LL.

Parsing SQM and boltwood files to find the files with smallest time deltas

We need to put the boltwood in the correct format and manually edit the output of the script to a time frame near our observations, or else matching will take forever. This part could be improved. Then we parse the boltwood and sqm timestamps and calculate the tdelta from each fits file, the smallest tdelta will be assigned with the matching SQM and boltwood info.

1. Use **nsb_boltwood_logger.py**
    a. nsb_boltwood_logger.py NAME OF BOLTWOOD FILE
        i. Creates boltwood.txt
2. Delete the rows far away from the timeframe of observation in boltwood.txt
3. Use **nsb_parser.py** to match information
    a. `nsb_parser.py nsb_data.csv boltwood.txt sqmtel.txt sqmzenith.txt`
        i. This will take some time
        ii. Creates
            1. nsb_bolt.csv
            2. nsb_sqm_tel.csv
            3. nsb_sqm_zenith.csv
        iii. These files have the matched fits files with measurements

Plotting the data

Now that we have all the required information, we can plot the data.

1. Use **nsb_plot.py**
    a. This has many options to plot the data. Use -h or --help to learn more about the options or read the code.
    b. You might need to manually edit nsb_plot.py to change the magnitude range (y-axis) to better fit the data.

Inject the sqm, bolt, and nsb data into the RAW RAW RAW files for LL

Remember that raw folder with all the raw V fits files, well copy that into a raw folder called LL_raw.

1. Move nsb_data.csv, nsb_sqm_tel.csv, nsb_bolt.csv, and nsb_sqm_zenith.csv to LL_raw
2. Use **nsb_inject.py**