This entire map describes the class 'fakeSNplanter'

initially we will write each colored set of tests and modules as separate python modules. When a module passes the unit tests and is ready for other developers to work with, it should be adopted into the class as a class function.

diff image test data: a single fits file from LCO static sky image test data: a single fits file, a single-night image (the one used to make the test diff image) template image test data: a single fits file, a deep stack (the one used to make the test diff image)

galaxy catalog test data: a catalog as an ascii text file, initially empty pipeline log file: for recording progress through the stages of this pipeline. an ascii text file, initially empty

test data level 0: pristine

Module: Build an ePSF model

- identify stars in the static sky image by making a query to the online Gaia database
- build an ePSF model from those stars, add it as an extension to the input fits image and save the modified fits image to disk
- record pre-existing info about the image + measurements of the ePSF model in the image database: FWHM, zeropoint

in each module, add a switch: writetodisk=True for testing, False by default otherwise

Fits image Unit test:

- check that each modified image (diff im + static sky) has an extension with the ePSF model
- check that the ePSF model has a non-zero flux

Pipeline log Unit test:

- check that there is one entry in the logcorresponding to the test diff image
- check that all required entries in the log are present, and nothing more: image name,

test data level 1: has psf model

Module: Plant fakes

- using the ePSF model embedded in the fits file, plant a grid of fakes or plant fakes around galaxies with varying magnitudes (fluxes), mimicking strong-lensing sources
- write info into the fits headers for each planted fake, including the location and total flux
- write out the modified image with fakes planted as a new fits file
- record in the image db that fakes have been planted in the image

Fits image Unit test:

- subtract the unmodified image (level 1 or level 0) from the modified image (level 2)
- check that the sum of all pixel values in the resulting diff image is equal to the sum of all fake fluxes from the header

Pipeline Log Unit test:

- check that the image with fakes added is marked as such in the image DB.
- (future: when implementing fakes that have a full SN light curve, will need to record something about that in the image DB)

test data level 2: has fakes added

Module: Detect sources in the diff image

- use an astropy threshold detection algorithm to identify transient source candidates in the diff image fits file
- record the locations and fluxes of candidate sources in an external source catalog file
- if a fake is detected, mark it as such in the source catalog
- if a fake is not detected, add it to the source catalog (as a false negative)
- run aperture photometry on each fake source

Flts image + Source catalog unit test

- check that the source catalog exists
- check that all fakes in the fits image header are also marked in the source catalog
- check that the measured flux of each fake matches the input flux (giving a LARGE allowance for error)

test data level 3: detections

Module: Utilities

all functions of general use should be put here e.g., read/write tasks, updating log files, anything that is not specific to a particular stage of the processing pipeline