DEFINE function (importobs) ##import the csv file containing radius, Teff, photons, days data

READ csv file

IDENTIFY 4 data columns

CREATE DataFrame with this data

PRINT DataFrame

CONVERT day column from 1/10 days to julian date

CALCULATE luminosity from photons data

END importobs

DEFINE function (makeobs) ## flag rows of data according to how ROTSE takes observations (identifying the epochs where observations are actually made)

## Variables:

## Teff= effective temperature output from MESA/phot\_per\_sec

## Photons = photons emitted by the star output from MESA/phot\_per\_sec

## days= age of the star in days output from MESA/phot\_per\_sec

## possibleobs= list that contains the possible observations from MESA/phot\_per\_sec. creates a 10 year lightcurve/list of possible observation epochs (epoch\_list)

## time=1/10 days converted to Modified Julian Date

DEFINE function (possibleobs) ## creates a 10 year lightcurve/list of possible observation epochs

INPUTS: photons, time

OUTPUTS: fullphotons, fulltime

END possibleobs

DEFINE function (weather)

## n=random number generator 1 output: number of instances of bad weather

## n2=random number generator 2 output: duration of each instance of bad weather

## Newphotons: updated list of photon data with flagged points from to previous function

## Newtimes: updated list of time data with flagged points according to previous function

INPUTS: fullphotons, fulltime

IMPORT Numpy for random number generator

DEFINE function (badweather)

INPUT: fullphotons, fulltime

RUN random number generator for n of instances of bad weather

OUTPUT: random number, n ##selects n random sections of data for instances of bad weather

END badweather

DEFINE function (duration)

INPUT: random number n

RUN 2nd random number generator for how long bad weather lasts

OUTPUT: random number, n2 ##select n2 data points to flag after initial n

END duration

SELECT n number of random rows and n2 number of rows after that initial selection to flag for bad weather

OUTPUTS: newphotons, newtimes

END weather

DEFINE function (daynight)

## date=date of observations of interest in modified julian days

##latitude=latitude of observation location

##longitude=longitude of observation location

##nighthours=number of hours per night (window between sunrise and sunset)

INPUTS: newphotons, newtimes, date (in MJD), longitude, latitude

DEFINE function (night) ##Calculate how many hours of night time during that time of year

INPUTS: date (in MJD)

FORMULA: cos(wt)=tan(phi)tan(D)

OUTPUTS: (nighthours) number of hours per night (window between sunrise and sunset)

IF statement that defaults to 12 hours of day/night time if time of year not specified

FLAG (number of hours per night/24 hours) rows of data

OUTPUTS: newphotons2, newtimes2

END night

END daynight

DEFINE function (abovehorizon)

## Right ascension= RA of the cepheid of interest

## Declination= DEC of cepheid of interest

## elevation= elevation above the horizon of cepheid at the location and time of interest

## Newphotons3, newtime3: information to be plotted to create new lightcurve

INPUTS: newphotons2, newtimes2, RA, DEC, latitude and longitude, date

DEFINE function(elevation)

INPUTS: RA, DEC, location of telescope

FORMULA: sin-1(altitude)=sin(DEC)sin(latitude)+cos(DEC)cos(h)cos(latitude)

OUTPUTS: elevation above horizon

END elevation

FLAG (number of data points) - (when above the horizon) rows

IF statement that defaults to flagging ~½ data if no RA and DEC are specified

OUTPUTS: newphotons3, newtimes3

END abovehorizon

Plot remaining data (luminosity vs time) must be able to plot in pyplot and gnuplot

END makeobs

DEFINE function (outputobs)

CREATE output csv file

CLOSE file

SAVE to directory

SAVE plot to directory

END outputobs