Quick install

By default, git is installed in all linux & mac systems.

To download pythia:

1. Go to https://github.com/colej/pythia
2. Click on the big green button that says Code
3. Copy the URL: https://github.com/colej/pythia.git
4. Using the command line:
    a. Go to the folder where you store your python libraries
    b. Enter: *git clone https://github.com/colej/pythia.git*

I advise that you use an environment manager, such as anaconda or preferably miniconda. Miniconda is a much smaller download and is usable for windows, mac and linux. Make sure you use python > 3.7! After you install miniconda (or anaconda) and append the conda pathway to your PATH variable, you can proceed with setting up the environment for pythia.

I have provided an environment file so that all you have to do is pass this along to conda and it does the rest. Before you use this, however, you have to change one thing in the pythia.yml file. At the bottom, the prefix variable currently says: /YOUR/PATH/TO/miniconda3... This needs to be changed to reflect the location of miniconda3 (or anaconda3) in your directory structure.

Once this is changed, save the document and enter: *conda env create -f pythia.yml*

This will probably take a while, so, have fun waiting! But, don't forget to hit yes to the prompt when the environment finally solves.

The code doesn't need to be compiled or anything – it should run smoothly right out of the box! All you have to do is ensure that the **pythia** folder is located somewhere that your PYTHONPATH variable knows to look.

That should do it! The script test.py gives an example of how to use the code in a rough sense, but you should always play around to get a better feel for it.

Functions:

pythia.timeseries.periodograms.LS_periodogram:

@input: **times** -- 1D array of time values
@input: **signal** -- 1D array of flux or magnitude values. The program does not automatically remove the mean!
@input: **f0** -- starting frequency. This defaults to the frequency resolution
@input: **fn** -- ending frequency. This defaults to the nyquist frequency
@input: **oversampling factor** -- This defaults to 10, so that df = f_{res} / **oversample factor**
@input: **normalisation** -- normalisation factor. This defaults to amplitude.

@output: **nu** -- frequency array. Ranges between f0 and fn in steps of df. Has units of 1 / time
@output: **amplitude** -- amplitude array. Amplitude of sampled function. Has units of signal.

pythia.timeseries.iterative_prewhitening:

@input: **times** -- 1D array of time values
@input: **signal** -- 1D array of flux or magnitude values. The program does not automatically remove the mean!
@input: **yerr** -- 1D array of flux uncertainties.
@input: **maxiter** -- maximum number of frequencies to extract, if SNR criterion not reached
@input: **t0** -- Reference date. If not provided, it defaults to the midpoint of the times array.
@input: **f0** -- starting frequency. This defaults to the frequency resolution
@input: **fn** -- ending frequency. This defaults to the nyquist frequency
@input: **df** -- frequency step. This defaults to 1/7 * the frequency resolution
@input: **snr_stop_criteria** -- Breaks the loop when a peak with signal-to-noise (SNR) of that peak is less than this value. Defaults to 4.
@input: **order_by_snr** -- Boolean. If True, the extraction proceeds according to the peak with the highest SNR, which is not necessarily the same as the peak with the highest amplitude. Defaults to False.
@input: **use_snr_window** -- Boolean. If True, calculates the SNR of a peak according to the noise level within +/- **snr_window**. If False, it uses the average noise calculated within **snr_range** Defaults to True.
@input: **snr_window** -- Window size in units of frequency, over which to calculate the SNR. Defaults to 1.
@input: **snr_range** -- Range over which to calculate noise if **use_snr_window** is False.

@output: **residuals** -- 1D array of fluxes or magnitudes after sine-model has been subtracted.
@output: **offsets** -- 1D array of fit offset values. Units of signal.
@output: **frequencies** -- 1D array of extracted and fit frequency values. Units of 1 / time.
@output: **amplitudes** -- 1D array of extracted and fit amplitude values. Units of signal.
@output: **phases** -- 1D array of fit phase values. Units of radians.
@output: **stop_criteria** -- 1D array of extracted SNR values.
@output: **noise_level** – 1D array of the noise level (as a function of frequency) estimated at the end of the routine