Semester I 2025

Astroinformatics I

Graded Practice 4

José B. Batista M.

1. Create a GitHub repository where you will at the end submit your graded practices so far, as well as today's graded practice.

The GitHub repository for all graded practices, including this one, is available at https://github.com/josebatistam/Astroinformatics_I. This repository is structured to organize all assignments submitted for the Astroinformatics I course. It serves as a central hub for the course work, ensuring easy access for review. The repository's layout is designed for clarity and maintainability:

- The root directory contains the main **README.md** and serves as the entry point for the entire collection of practices.
- Individual graded practices are organized within their own dedicated subdirectories, named / Practice_1/, / Practice_2/, / Practice_3/, and / Practice_4/.
- Each practice subdirectory typically follows a consistent internal structure, containing:
 - /codes: Houses shell scripts (.sh) and Python scripts (.py) relevant to the practice's tasks.
 - /latex: Contains the main Latex file for the practice (e.g., Practice_X.tex), along with auxiliary files generated during compilation (.aux, .log, .out, etc.).
 - /csv, /fits, /lc, /lightcurves, /lists, /plots: Specific directories for input data (like .csv or .fits files), processed light curve files (.lc), generated plots (.pdf), or intermediate lists, depending on the requirements of each practice.
 - Practice_X_Instructions.pdf: The problem statement or instructions for the specific practice.
 - Practice_X.pdf: The compiled PDF solution for the practice.
- A /Project/ directory is dedicated to the final project presentation, which synthesizes the work from the four graded practices.

The **README.md** file at the root of the repository serves as the primary entry point for anyone visiting the project page on GitHub. It provides a concise overview of the repository's purpose, its structure, and relevant information for users and maintainers. Key sections within this **README.md** include:

- **Project Purpose:** A clear statement on what the repository contains (graded practices for Astroinformatics I).
- Repository Structure: A brief explanation of the directory layout, detailing the organization of individual practices as outlined above.

- **Practices:** A listing of all submitted practices, with direct links to their respective files or folders within the repository for easy navigation.
- **Project Presentation:** Details about the final project presentation.
- Course Information: Details about the course, institution, and semester.
- Author: My name, as the creator and maintainer of this repository.

As an important note regarding the repository's contents, the local course directory was restructured for easier navigation and file access during my work. As a consequence, all the scripts (.sh and .py) and LTEX files in the /Practices/ subdirectory had to be updated to reflect these changes in order to prevent compiling errors. However, for transparency purposes, any code presented directly within the LATEX files in the submitted solutions has not been updated to reflect these modified scripts, preserving the original practice solutions as initially developed.

2. Write some basic documentation for how you processed the TESS light curves.

The documentation is available as a README.md file located in the /Practice_3/ directory of the GitHub repository. This README.md provides a comprehensive overview of the practice_3.py script's functionality. It details the step-by-step methodology, including:

- The process of loading and preprocessing TESS light curve data.
- Methods for identifying and handling outliers.
- How raw light curves are plotted and visualized.
- The implementation of phase-folding and Lomb-Scargle periodogram analysis for periodicity detection.

It also highlights that the Python script itself is well-documented internally through the use of docstrings for its functions, providing clear explanations of their purpose, arguments, and return values directly within the code.

The documentation also lists the necessary Python dependencies (which are given in the file requirements.txt) and highlights important considerations regarding periodogram interpretation, as discussed in the original practice solution. For detailed explanations, code usage, and dependency information, please refer directly to the README.md file in the specified repository path.

3. Identify some test cases for the processing of the TESS light curves and write them down (no implementation needed).

Given below are some test cases for the practice_3.py script, for each function.

Test Cases for get_lc_data:

Valid Standard File:

 Scenario: Provide a valid path to a standard TESS light curve .1c file with complete TIME, PDCSAP_FLUX, and PDCSAP_FLUX_ERR columns. Expected Result: The function should successfully load the file, return an astropy.timeseries.TimeSeries object, and correctly convert BTJD to JD and create DATE strings.

• File Not Found:

- Scenario: Provide a filename that does not exist in the lc_data_folder.
- Expected Result: The script should catch the FileNotFoundError and exit gracefully with an informative message.

Invalid File Content:

- Scenario: Provide a file that exists but is not a valid light curve format (e.g., a corrupted .lc file, or a text file with incorrect delimiters/headers), or a file with non-numeric flux/time data.
- Expected Result: The script should catch parsing errors (e.g., from type conversion) and exit gracefully with an informative error message.

Test Cases for identify_outliers:

Standard Data with Outliers:

- Scenario: Provide typical flux and dates arrays that include a few clear outlier points.
- Expected Result: The function should correctly identify and mark only the actual outlier points in the returned boolean mask, without flagging valid data.

• No Outliers / Uniform Data:

- Scenario: Provide flux and dates arrays where all flux values are very consistent (e.g., from a constant star) or where there are no points outside the threshold.
- Expected Result: The function should return a mask where all values are False (no outliers identified).

Test Cases for plot_raw_lc:

Successful Plot Generation:

- Scenario: Provide valid filename and lc_data with some identified outliers.
- Expected Result: A PDF file should be successfully created in the plots/ directory, displaying the raw light curve, correctly highlighting the outliers, and including all specified labels and legends.

Light Curve with Gaps/Missing Data:

- Scenario: Provide lc_data that contains NaN values or significant gaps (which would be handled by dropna() in get_lc_data).
- Expected Result: The plot should accurately represent the available data, showing the gaps where data is missing, and not generating errors due to the missing points.

Test Cases for fold_lc:

Known Periodic Signal:

- **Scenario:** Process **lc_data** from a star with a well-known, strong periodic signal (e.g., an eclipsing binary or pulsating star).
- Expected Result: The Lomb-Scargle periodogram should clearly identify the correct period as the highest power peak, and the phase-folded light curve should show a coherent, well-defined folded shape.

No Apparent Periodicity:

- Scenario: Process lc_data from a non-variable or very low-variability star.
- Expected Result: The periodogram should show low power peaks, indicating no strong periodicity, and the phase-folded light curve should appear flat or noisy without a clear folded shape.

• Different Period Ranges:

- Scenario: Vary min_period and max_period to constrain the period search (e.g., a very narrow range around an expected period, or a very wide range).
- Expected Result: The periodogram should adapt its search space, and the best period found should be consistent within the search range.