PHYS265 2025 Code Project - version 14-may

See also: https://github.com/astroumd/PHYS265-spring25-final/blob/main/code_project.md for updates

This *Code Project* is your final project for PHYS265, there is no exam. In this project you will select a (python based) open source project. It is your task to evaluate this software package by using it and writing a report (software review if you wish) based on an itemized list you see below. Describe them in your own words and experiences, the intent is not to reproduce their documentation.

Project titles are due by Wednesday May 7. Two more in-class `lab` sessions are on May 9 and May 12, with final office hours on May 15. **Due date will be Sunday, May 18, at the usual 11.59pm time**

Email your project title to: teuben@umd.edu - you will need to have received an approval from Peter or Brian before you can start work.

You will first need to learn how to install and use the package, show some results and one or two figures to illustrate your findings. All of this will be summarized in an itemized report presented in PDF. You also need to show your code and data example(s) how you exercised the code, in a standard jupyter notebook (ipynb file). A README file is optional. Please name and date your report!

NOTE: We again submit on github, and you will use the **Project** folder inside your existing github repository that you used for the labs.

You are welcome to suggest your own project too. Either way, discuss with your instructors which package you choose and you may get additional guidelines. You can find some suggestions in our list below extracted from ASCL and JOSS.

Some packages may not work well on your particular computer. Some require extra software, which might make it too complex to install, or require a supercomputer. Always run your selection by your instructors for approval. We do not want you to waste time getting the package to run.

Your report can be an enumerated list of answers (see also the NEMO example. Each question below is worth 3 points, but final score is normalized to the questions that have an answer (e.g. if a package has no citation). You are however responsible for finding out that there is no answer!

1. Title: your title should start with the package name, feel free to use a Combination Title, e.g. NEMO: where did the fish go? Author: don't forget to author your work, a date is always nice too.
2. name of the package, describe what is the basic aim of what the package does or solve?
3. why/how did **you** select this package?
4. how old is the package? does it have a geneology, i.e. what related codes came before or after. are there other codes you can find that solve the same problem? Can you figure out which version you installed? Often this is

   ```
   import package; print(package.__version__)
   ```

5. is it still maintained, and by the original author(s)? Are there instructions how to contribute to this project.
6. evaluate how easy it was to install and use. What commands did you use to install?
7. does it install via the "standard" pip/conda, or is it more complex?
8. is the source code available? For example, "pip install galpy" may get it to you, but where can you inspect the code?
9. is the code used by other packages (if so, give one or two examples). ASCL codes have citations via their ADS link. See also 22.
10. How is the code used. Is it commandline, python script, or a jupyter notebook, or even a web interface?
11. provide examples using the code. if you prefer to use a jupyter notebook instead of a python script, that's ok. See also 12.
12. does the package produce figures, or are you on your own? Is matplotlib used?
13. your code and report should show at least one figure, and create a nice figure caption explaining what it shows. You notebook should show how the figure was made (i.e. be reproducable). Second figure is optional, but only use it when you need to illustrate something extra. This implies you may need to add any data you need to your repository!!! NEW
14. is the package pure python? or does it need accompanying C/C++/Fortran code?
15. what is the input to the package? Just parameters, or dataset(s), or can they be generated from scratch?
16. what is the output of the package? Just parameters, or dataset(s)?, or just a screen output you would need to capture
17. does the code provide any unit tests, regression or benchmarking?
18. how can you feel confident the code produce a reliable result? (see also previous question)
19. what (main) python package(s) does it use or depend on (e.g. numpy, curve_fit, solve_ivp) - how did you find this out?
20. what kind of documentation does the package provide? was it sufficient for you?
21. if you use this code in a paper, do they give a preferred citation method?
22. provide any other references you used in your report.
23. can you find two other papers that used this package. E.g. use ADS citations for ASCL based code. See also 8.
24. did you have to learn new python methods to use this package? Or was the class good enough to get you through this project.
25. Final Disclaimer: you need to state if you have prior experience in using the package or the data, or this is all new to you. In addition, if you collaborated in a group, as long as this is your work.

Summarizing your github submission should contain the following files:

1. The PDF report with 25 answers, with at least one figure

2. A notebook showing how you used the code, and in particular how your figure was made
3. Any optional datafile(s) needed to run the notebook, to make your project reproducable.
4. An optional README file if you feel you want to explain how to run your code. You can also include this as a cell in the notebook.

Suggested projects

This list below is an extraction from https://ascl.net/code/search/python (a full list returns well over 600 codes). Another option would be to browse JOSS (https://joss.theoj.org/papers/search?q=python) the Journal of Open Source Software, also a sizeable list, but none of those are reported below.

There is a new AI driven interface to ASCL, you can explore this AstroCoder project here: https://nolank.ca/astrocoder as wel as read more about it on the authors blog on https://blog.nolank.ca/astrocoder

Codes annotated with **[peter]** are codes that Peter uses from time to time.

- PDRT: Photo Dissociation Region Toolbox - https://ascl.net/1102.022

  OK. This package was written by two astronomers at UMD.

- PySpecKit: Python Spectroscopic Toolkit - https://ascl.net/1109.001 **[peter]**

  OK. Fitting spectral lines. see also specutils.

- PyEphem: Astronomical Ephemeris for Python - https://ascl.net/1112.014

  OK. but has no graphics, you will need to create your own

- EzGal: A Flexible Interface for Stellar Population Synthesis Models - https://ascl.net/1208.021

  OK, but no example fits file?

- pNbody: A python parallelized N-body reduction toolbox - https://ascl.net/1302.004

  A general “pip install pnbody” did not work, but downloading from github and using

  ```
  pip install ./pNbody
  ```

  worked for me.

- corner.py: Corner plots - https://ascl.net/1702.002

  OK. Together with emcee a very popular package in the literature. Explores the covariant matrix graphically.

- emcee: The MCMC Hammer (possibly most used code) - https://ascl.net/1303.002

  Markov chain Monte Carlo (MCMC) Ensemble sampler - probably one of the most downloaded codes used in papers. Together with the corner.py code, which plots up the covariance between variables.

- Aegean: Compact source finding in radio images - https://ascl.net/1212.009

  Although python, more command line driven code

- Galactus: Modeling and fitting of galaxies from neutral hydrogen (HI) cubes - https://ascl.net/1303.018

  complex build system, needs c++ and boost.

- Astropy: Community Python library for astronomy - https://ascl.net/1304.002

  OK. large body of code, some subset should be used. There’s a nice model fitting tool in astropy. also specutils, which is like pyspeckit. And of course fits I/O where you can learn about FITS cubes.

- SunPy: Python for Solar Physicists - https://ascl.net/1401.010

- Gammapy: Python toolbox for gamma-ray astronomy - https://ascl.net/1711.014

  Needs data.

- galpy: Galactic dynamics package - https://ascl.net/1411.008 **[peter]**

  OK. may need compiler, not always ideal for windows

- PyBDSF: Python Blob Detection and Source Finder - https://ascl.net/1502.007

  Needs fortran and boost, and some python packages. Source code has test image. Invented their own “ipython” shell

- pYSOVAR: Lightcurves analysis - https://ascl.net/1503.008

very author specific

- PyTransit: Transit light curve modeling - https://ascl.net/1505.024

  OK

- pyro: Python-based tutorial for computational methods for hydrodynamics - https://ascl.net/1507.018

  Regular pip install didn't work, but from source it did.

- SavGolFilterCov: Savitzky Golay filter for data with error covariance - https://ascl.net/1601.012

  Just a code from a paper. no pip.

- POPPY: Physical Optics Propagation in PYthon - https://ascl.net/1602.018

  OK

- Lmfit: Non-Linear Least-Square Minimization and Curve-Fitting for Python - https://ascl.net/1606.014 **[peter]**

  ok, good alternative to curve_fit(). Here it would be interesting to show how you can fit something with curve_fit() and lmfit. Your instructors may give you a dataset to fit and compare results.

  One possibility for data is the paper "The Legacy of Henrietta Leavitt: A Re-analysis of the First Cepheid Period-Luminosity Relation"

  https://arxiv.org/abs/2502.17438

- PRECESSION: Python toolbox for dynamics of spinning black-hole binaries - https://ascl.net/1611.004

- Gala: Galactic astronomy and gravitational dynamics - https://ascl.net/1707.006 **[peter]**

  OK

- PROFILER: 1D galaxy light profile decomposition - https://ascl.net/1705.010

  Seems old by now. Indeed, it's python 2 only

- SETI-EC: SETI Encryption Code - https://ascl.net/1803.009

  A serious challenge!

- pydftools: Distribution function fitting in Python - https://ascl.net/code/v/1940

  OK

- Photon: Python tool for data plotting - https://ascl.net/1901.007

  A regular "pip install" claimed versions were not matching, but a manual "pip install -e ." in the source code worked, but then complains that my matplotlib has no mplDeprecation attribute.

- oscode: Oscillatory ordinary differential equation solver - https://ascl.net/1908.012

  builds a wheel; mimics solve_ivp()

- GWpy: Python package for studying data from gravitational-wave detectors - https://ascl.net/1912.016

  ok

- seaborn: Statistical data visualization - https://ascl.net/2012.015 **[peter]**

- Citlalicue: Create and manipulate stellar light curves - https://ascl.net/2202.014

  Lot of dependancies that were not listed (arviz, celerite, corner,. ...)

- CosmosCanvas: Useful color maps for different astrophysical properties - https://ascl.net/2401.005

  OK

- CONCEPT: COsmological N-body CodE in PyThon - https://ascl.net/2306.035pnbody

  Too compute intensive!!!

- AMUSE: Astrophysical Multipurpose Software Environment - https://ascl.net/1107.007 **[peter]**

  Most likely too large and complex, it's a python package talking to legacy codes to run a set of simulations of different nature (n-body, stellar evolution, hydro dynamics etc.) Has a sister code called OMUSE from our Oceanography colleagues

  Needs MPI and various support libraries.

- FINUFFT: Flatiron Institute Nonuniform Fast Fourier Transform - https://ascl.net/2412.007

Has CPU options, GPU is over our paygrade. Nice way to compare good old numpy fft with something new?

- McFine: Muli-component hyperfine fitting tool - https://ascl.net/2411.021

  multi-component hyperfine spectra fitting in astronomical data. MCMC approach. Can be related to PySpecKit.

Example datasets

These will be in the subdirectory PHYS265-spring25/data. Check the README file for potentially updated descriptions.

- a fits cube: NGC6503.fits
- an ascii spectrum with spectral lines to fit:

Installation Guidelines

Most python packages can be installed with pip. Within JDL this would be the following command (but check the README file in the package for guidelines, sometimes a conda solution is given as well), e.g.

```
!pip install galpy
```

Some packages will rely on you having a C compiler, and if that fails, it is better to find another package before spending too much time solving compiler and linker problems. There is also the danger that mixing **conda** and **pip** may mess up your python environment.

If you have downloaded the source code via git, you can also install it as developer, i.e. while you would be editing their files, your modified code would be used! In a terminal shell this would mean the following commands

```
git clone https://github.com/jobovy/galpy
pip install -e galpy
```

There were a few packages where the regular pip install did not work, but a source install like this did work. go figure.