Pre-class assignment #5

PHY-905-005
Computational Astrophysics and Astrostatistics
Spring 2023

This assignment is due the evening of Wednesday January 25, 2023. Turn in your code and all materials via the GitHub Classroom.

"To err is human; to really foul things up requires a computer."
– Bill Vaughan (columnist) [1]

Numerical Linear Algebra Reading:

1. Sections 6.2-6.4 of *Computational Physics*, by M. Newman. **NOTE:** Section 6.3 is long and complex – just skim that to get the general idea, and focus on Sections 6.2 and 6.4!
2. Section 5.4 of *An Introduction to Computational Physics*, by T. Pang (optional; PDF provided)
3. NumPy linear algebra routines (reference)
4. SciPy linear algebra routines (reference)

Debugging Reading:

1. "Debugging truths" (included PDF)
2. Debugging Strategy - Patricia Shanahan, Association for Computing Machinery. Note that if this website seems to be non-functional, you can look at a saved version on the Internet Archive
3. Documentation for pdb – the Python debugger. Also read this short but useful pdb tutorial

Some potentially helpful debugging resources:

1. A handy reference sheet of pdb commands
2. Python Tutor, a tool that graphically visualizes the line-by-line execution of a piece of code.
3. ipdb, the Python debugger for IPython, which is a more feature-rich Python interpreter
4. JupyterLab Debugger documentation for the JupyterLab notebook user interface.
5. pudb, the full-screen, console-based debugger for the true command line fanatic.
6. VSCode Python debugger, the Python debugger module for Visual Studio Code (which is a really awesome and useful integrated development environment, or IDE)

Note: additional material is on the next page!

[1] This is not precisely the version that your professor first learned, but the language is more appropriate for polite company.

Coding assignment:

Consider the following pair of functions:

$$f_1(x, y) = x^2 e^{-(x^2+y^2)/4} \tag{1}$$

and

$$f_2(x, y) = 4x^2 + 1.5y^2 \tag{2}$$

which can be found in the file `func.py`.

Implement the **multivariable Secant method** (not Newton's method - calculate the derivative numerically, modifying code you've written in the past!) to find the joint root of f_1 and f_2 (which can be found by inspection to be at $x = y = 0$). Choose a starting point that is near zero but not precisely at zero (say, $x = y = 1$), and move progressively further away. How does that impact convergence and number of iterations? Compare this to the `root` method in the `scipy.optimize` package. How do your results compare, particularly as you experiment with different methods in `root`?

Handing it in: Submit your *easy-to-read and commented* code to the repository (using the class coding standard as a template). Also, put any remaining questions that you might have about the linear algebra readings or this particular assignment in the file `ANSWERS.md`.

Debugging assignment:

1. A simple but broken code has been included in the file `string_reversal.py`, with a Jupyter Notebook version in the file `string_reversal_notebook.ipynb`. Edit this code to add python break points and then use python or IPython at the command line, or the JupyterLab debugger in JupyterLab, to step through the code to identify the broken piece of code (following the suggestions in the code's comments as to what you should print out, where you should put break points, etc.) When you fix the code, what happens?

2. Thinking about the two readings on debugging, what ideas/suggestions seem the most useful for you? Which ones seem the hardest to implement in your day-to-day programming?

3. Do you have a favorite debugging tip or trick that wasn't in the readings? What about a favorite debugger that's part of an editor or Integrated Development Environment (IDE)? Please share it in `ANSWERS.md`.

4. Do you have any additional questions, comments, or concerns about debugging? Please share it in `ANSWERS.md`.

Handing it in: Include your modified code and your answers to the questions (in the file `ANSWERS.md`) in your assignment.