CMEE Masters: Computing Coursework Assessment

Assignment Objectives: To work on a series of computing/programming exercises and problems in a coherent, modular, reproducible workflow under version control.

Note that:

- *The overall assessment will typically have significantly lesser marks than a simple weighted average of each week's points because the overall assessment is based on not just the "Computing Coursework Assessment Criteria", but also the the "Marking Criteria for Exams, Essays and Coursework". Both sets of marking criteria are in the Assessment Appendix of the online TheMulQuaBio notes and git repository.*
- *In your 1:1 post-assessment feedback session, we will discuss where you gained or lost marks, and what you could have improved further. To the extent possible, please come with questions about specific scripts based upon the overall and weekly feedback you have received. This may require you to compare your code with the solution code in many cases.*

Student's Name: Cherie Yu

1 Specific feedback

1.1 The Good (what you did well!)

1. Found all the core CMEE weekly directories in your parent directory.
2. Your Git repo size when I checked week 7 was about 8 MB – nicely compact! This suggests you correctly suppressed unnecessary files from version control, and did not commit excessively. It could also mean that you did not commit enough, and/or somehow along the the way lost parts of your git history – but we don't check these possibilities!
3. You have included both overall and week-specific readme files. These are comprehensive and well-organised, and you have included things like dependencies for each one (though the installation instructions for each package was perhaps a tad unnecessary!) Also check out this resource: `https://github.com/jehna/readme-best-practices`. As you become a seasoned programmer, you will learn to make the readme file descriptions even more informative yet succinct.
4. You had a .gitignore throughout, with meaningful exclusions specific to certain weeks. Good! If you wish to fine-tune exclusions further, you will likely find this useful: `https://www.gitignore.io`.
5. Your Python is generally nicely modular – this is nice and Pythonic!
6. You have done a good job of the coding overall, but are occasionally let down by not having noticed and/or addressed some easily fixed errors. Attention to detail is critical as a programmer, since even minor errors can mean that your script doesn't run at all (or fails at an inopportune moment).
7. Your Groupwork practicals were all in order, and your group did well in collaborating on it. More feedback on this in the 1:1 sessions.

1.2 The Bad (errors, missing files, etc)

1. `basic_io1.py` threw an IndentationError because one of your docstrings was not indented to the same level as the `with` statement immediately before it. Similarly, `loops.py` threw an IndentationError because the docstring for the final while loop has a different indentation level than the print() command on the following line. It's important to be aware that in Python, docstring indentation can cause errors just like the indentation of any other expression!

2. `LV1.py` threw a syntax error due to your docstring ending with an extra ". This kind of mistake is easy to make when writing the same character multiple times, but it is critical to be vigilant and to run and test your code once you're done writing it, so that you can be sure to catch fatal but trivial errors such as this.

3. Although your file organisation is generally neat, some of your script outputs (e.g. histogram.pdf & keywest.pdf) have been saved to your Week3/code/ subdirectory, rather than placed into the relevant results subdirectory. This is messy, and will make it hard for you and/or users of your code to find things within your project structure, particularly when it comes to more complex projects.

1.3 The Ugly (niggling issues like commenting, cosmetics, complexity of code, etc)

1. Commenting could be improved – you are currently erring on the side of overly verbose comments, which is nonetheless better than not commenting at all, or too little! This will improve with experience, as you will begin to get a feel of what is "common-knowledge" among programmers, and what stylistic idioms are your own and require explanation. In general though, comments should be written to help explain a coding or syntactical decision to a user (or to your future self re-reading the code!) rather than to describe the meaning of a symbol, argument or function (that should be in the function docstring!).

2 Overall Assessment

Overall a rather good job. Although a small number of your scripts did retain some fatal errors, most of them ran without issue. Try to be a little more vigilant in chasing down errors in future. Your commenting is very thorough, perhaps a little too much so, but this is also likely to be a tendency that fixes itself with experience. A solid job overall, well done.

Provisional Mark: 68%

Signed: Alexander Kier Christensen & Samraat Pawar

April 13, 2022