Last updated: 9/1/20

Math 260: some course guidelines Fall 2020

1 Collaboration

It is essential to understand what it means for code to be 'your own' work, so that you can freely collaborate with others on homework without fear of crossing the line into plaigarism.

I want to emphasize that that **collaboration is valuable** for coding assignments, and can often give you far more insight and productivity than working alone. For this reason, I hope that fear of 'accidentally' copying too much does not deter you from sharing work.

Always strive to write code that can be read by someone else (not just yourself). Having an actual person read your code introduces pressure for you to write better, which improves your coding habits (moreover, in the way you need for group programming in the real world).

It does, however, take a bit of work - you'll need to have a good way to share and discuss code (e.g. `github`, slack, etc.). This is worth the investment.

2 Attribution

It's useful to see how code is written from other sources. Material on scientific computing is plentiful, if a bit scattered, online. You are certainly welcome to make such comparisons, but be careful that what you find doesn't 'solve the problem' entirely - once you see a solution, it can be hard (and less productive) to write your own version. I recommend avoiding such code until *after* your work is done.

Looking up general concepts is fine (e.g. syntax for a while loop), or supporting calculations within an algorithm that aren't 'the point' of the assignment, but you are certainly **not** allowed to copy large blocks of code from other sources. If a nifty trick is borrowed from a source, attribute it in a comment. Code that is attributed is always safe, and if it is not a main part of the assignment, will not lead to any loss of points.

A warning: Code found online may be of dubious quality - be careful when using miscellaneous sources (e.g. stackexchange, random lecture notes). The material is typically still useful, but be wary (think before you use code).

3 General expectations for good code

This is the list of guidelines to follow that you are responsible for. As the course progresses, the guidelines will be expanded (we learn a new aspect of python, then after some practice good implementation is expected). As learning good coding practices is a primary goal, you should pay close attention to these guidelines.

Here are features of good code that should be implemented in your work:

1) **The code works:** The code runs as requested in the assignment. This has two components: (i) when run, the program produce the desired output and (ii) your routines process inputs in the expected way. (Check: if a random (valid) input were sent to your functions, would they behave as expected?)

- If your submitted code does not work, or has a bug that was not fixed in time, indicate this in a comment inline (and perhaps a brief note at the start of the file).[1]

2) **Labels are correct:** Your name and the date of submission appears in a comment on the first line of each submitted file. Files have names that are clear but **not too verbose**, typically one word like `fibonacci.py`).

3) **Reasonable comments:** Each major function has a comment with a brief description (about one line). Short functions typically do not need this. We'll discuss how to comment code, and after this is introduced (doc-strings) there will be further guidelines.

   Use comments on lines when you need to clarify a step, but **only as neede de**. Python code should mostly speak for itself (readable without comments).

4) **Good style:** Code should be clear and readable; we will discuss details for each aspect of the language, among other general style points. That is, you should incorporate the principles introduced in lecture into your code.

[1]Note that the best remedy here, when encountering a bug, is to discuss it with someone and try to fix it - a reason to start work on code early. Of course, sometimes time is limited, so submitting buggy code is understandable!

4 Guidelines on communication

I hope that the virtual space of the online class can mimic, as much as possible the community that develops in the actual classroom. There are two components to this:

4.1 Zoom

The zoom class is intended to be a classroom-like experience for a lecture. This takes a little bit of extra effort on your part to overcome the technological issues...

- If you have a camera, I encourage you to leave it on during class if able. You are **not required** to do so, however, if there is any reason it cannot be on.
- You can speak up in class at any time, but to do so you should first 'raise your hand' [in Zoom] and I will call on you. (Normally, I would be fine with a direct interruption, but in the virtual class, this step helps maintain the flow of class).
- Other Zoom tech: You will likely need to share your screen at some point. This can be done in Zoom either by sharing your entire desktop or one program. Note that if you are sharing code that outputs to a different program (e.g. code that makes a plot rendered in `gnuplot`), sharing your entire desktop may be the better option.
- I will be available via Zoom during the specified office hours, but I will also be, effectively, "in my office" at other times. If you want to meet during weekdays, it can be arranged by appointment; just send me an e-mail and I can set up the Zoom meeting.

4.2 Piazza

The Piazza site is the primary forum for out-of-class discussion.

- Ideally, some of the informal communication that typically occurs in person can be replicated through Piazza - consider it an opportunity to build a sense of community and collaboration in the course.
- Try to make an effort to start posting early to get in the habit. This may pay off later - for instance, in troubleshooting more complicated code later in the course.
- Anything related to the course can be discussed - just be thoughtful and respectful in posts. If you'd like a question answered, feel free to ask, no matter how small or narrow it seems (in programming, it's easy to get stuck on things that "should" be obvious).
- Piazza is a good place for posting programming questions - it's likely that other students may have had the same question and/or found an answer.
- Questions you have for me on the material should be posted to Piazza rather than by email (so that you can properly format math/code). Individual concerns (e.g. absences etc.) can be sent to me via email.