Post-Week 1 Notes

Well, we've survived our first week of Sociological Gobbledygook (and one room change—which we're keeping, BTW). We've learned the basic mechanics of programming, and, more importantly, we've worked on building a mental model for how talking to a computer works by completing a number of exercises.

For the in-class time in week 2, expect more of the same, lots of exercises—but now they're going to get a little bit more complicated. I'll also be pushing out the reading for week 3 soon.

By now, everyone should have a working Azure notebooks setup and a Github account; you should also have received an invitation e-mail to join DataCamp. I encourage you to take advantage of DataCamp beyond the stuff I've assigned: you have free access to all their lessons for the duration of the class period, but it usually costs $30/month, so milk it for all it's worth while you can. And let me know if there are any problems.

When I add new lessons, we'll have to talk a little bit more about how git works to enable you to pull the changes into your local version of them in Azure notebooks; please remind me to talk about that in class if I haven't gotten instructions posted by the beginning of next week...

Finally, a quick note: I've noticed a few people taking notes in the class, including snippets of code, in Microsoft Word. Word is not a great choice for writing code, because it saves in a format that mucks around with the text encoding (and also might screw you up by using things like autocorrect), so there's a good chance that you won't just be able to run code saved in Word anywhere else. I recommend using a professional-quality text editor ("text editor" is in opposition to "word processor," so also don't use Pages, WordPerfect, Google Docs, OpenOffice, etc.; text editors are also sometimes called "code editors.")

Another reason to use a good text editor is that they all have packages that you can use to extend their functionality, for example to give them the power to take Python code files and highlight different elements (so, for example, strings are in a different color), auto-complete variable names to avoid typoes, and such. Also, because they save in "plain text" format (you'll soon learn that there's really no such thing as plain text, but they save in as close as you need to plain text), you can use them to just create code files that you can send directly to a Python interpreter to run on your local machine. This is difficult to do in a word processor.

Here are my suggestions, and some notes on each of them:

- Atom is probably the best beginner-friendly free text editor. It's available for all major operating systems. The one downside is that (for programmer-laziness reasons that I'll explain below in case you're curious) it tends to be a big battery drain, so I wouldn't leave it running outside of class/without plugging in.

- VS Code is Microsoft's version of Atom—they're extremely similar (I suspect including similar battery drain problems, though, because of my visceral hatred of all things Microsoft I haven't used it for long enough to see myself), but some people prefer the one over the other. It's also totally cross-platform.
- For Mac users only, BBEdit is an excellent text editor with a free and a paid version, where the free version works totally fine for any uses you'll need. There's also Sublime Text which is actually by far the best Mac-specific editor (in my opinion), but is not free—it costs something like 70 bucks, although they will let you use it without paying so long as you're willing to put up with period annoying nag message pop-ups. Many people also use TextMate, but I've never tried it.
- For Windows-specific applications, most people seem to recommend Notepad++ which is free. I've never use it (though I think it's installed on some of our university computers), but it's very popular, so it's probably good. Apparently there's also a Sublime Text for Windows (which I just disovered); if it's stable it's doubtless amazing.
- If you're using Linux, you probably already have a text editor you like; if not, talk to me or try Gedit.

explanation of the battery drain problem: a lot of programmers who don't want to write separate code for different operating systems use a cross-platform framework called Electron to do it instead. Both Atom and VS Code are written in Electron. The problem with Electron is that it uses web browser technologies to get it to work on both platforms, so, essentially, it just runs a bunch of instances of Chrome in order to display documents. Chrome is a really heavy program that itself draws a ton of power, and Electron apps do it too. Incidentally, Slack is also an Electron app, so if someone's made you use that and you've noticed it nuking your battery, that's why.