Hello mentors! This is Will Tirone, a past Python mentor. I’m writing this guide alongside the syllabus to help you guide conversation and demos in the class. Obviously, feel free to deviate as you see fit to discuss whatever students are curious about, but these are my thoughts as I was making the syllabus. When I mentored, it was exclusively over zoom so that affects my thoughts on how to approach class, so my ideas may not necessarily carry over easily to an in-person setting.

I also developed an example project [HERE](https://colab.research.google.com/drive/1riRDLddFwWdfTx7o_a0cVcCD8sOTCaIn#scrollTo=NRTqvQXLb3n3) that you can follow along with and present each week of the class. I used topics from the corresponding week and worked through a data set with them. You can take this at whatever pace you want, but it’s set up to take place over 9 weeks. The only tricky part is that since this is on Google CoLab, the data isn’t readily loaded in. You could mount a google drive or download the notebook locally and run it if you really wanted to, but I would probably just walk through the cells and if you have to run new examples, just find another data set and make examples based on that. Spend about 30 minutes each week walking through each of the weeks in the project, and just use this as a guide for discussion. The students don’t have to do anything with this other than listen, this is simply meant as a guide so the mentors don’t have to spend an extra week writing code to show examples in class. I also (hopefully) have enough written instructions in there that if you don’t have time to make it through the whole thing, students should be able to read it on their own and understand the ideas. It’s also intended to show them topics from the syllabus while being an example of how they could meet requirements for their projects.

| **Week** | **Notes** |
| --- | --- |
| 1 | Your most important goal for this class is making sure everyone can run Python on their machine. You might have to pry a little bit, because the shy ones will ignore the question. It’s very easy on zoom to just hide away with your camera off, so you may have to ask those students in particular if they can run things. I have a link to install Anaconda so they can use Jupyter / Spyder, my preferences, but if they want to use VSCode or something else that’s totally fine. Some of them get hung up on choosing an IDE, but just reassure them that there are a million choices and as long as they can run Python it’s completely okay. Emphasize that Jupyter is popular because of the “interactive output” style of programming, and being able to display data / plot inline.  Go over the syllabus at a high level, answer any questions about the class. The first class is mostly very basic stuff about answering questions, telling them what to expect over the next few weeks, and introducing everyone.  On zoom, we would typically split into breakout rooms evenly with the mentors. We’ve done both random breakout rooms or assigning people to specific mentors and having those mentors checking in every week then talking to them in a breakout room. Honestly, either option can work depending on mentor preference but I lean towards just randomly assigning into breakout rooms and doing standup. Just do a quick 2-minutes per person “tell me about what you worked on, anything you had trouble with, and any project ideas”. That usually suffices, but if you have a more formal agile framework you want to use with them that’s fine too. |
| 2 | Start with standup / breakout rooms and ask them how they felt about the videos. At this point, people are probably pretty caught up but **in future classes people will start to fall behind on videos.** I have seen many very capable students just fall behind on the videos and drop out of classes (either due to other commitments or feeling like they can’t catch up) so try to gently emphasize to them the importance of staying on track. The most important goal of mentors is just to help keep them on track and keep at the material - you don’t have to be a code wizard, you just have to be encouraging to them. That’s so important, just keep encouraging them.  Answer any questions they have. At this point, some of them may realize that they actually don’t know how to run Python code and will ask for help, so definitely ask again if everyone was able to run code (even a simple hello world is fine). Some people may have attempted installing Anaconda (or whatever IDE) but are still having issues. It’s probably easiest to just hop into a breakout room with them and go over their particular issue and see if you can solve it. Sometimes they’re very easy to fix and sometimes they’re truly bizarre and we can’t figure them out.  This is also a good time to emphasize that if they have issues, their first step should be to Google the issue and **read the documentation first**. Sometimes that’s not too clear, so they can turn to Stackoverflow or something from there, and after trying that tell them to ask a mentor (or fellow student even). You don’t want to discourage them from asking you, but it will help develop their ability to solve problems by Googling them.  Encourage them to post questions in the main slack channel as well. This is also very important: some of them will be very hesitant to ask questions. Teachers everywhere have probably wondered how to engage students and get them to be more active in class for a millenia, but the same issue is present here. Just try to encourage active participation and tell them not to be afraid to just ask their question in the main channel so other people can see.  Discuss git. This is definitely something that you’ll go over for the remainder of the class here and there, but answer any questions about it. They had quite a bit to watch in terms of videos, so they’ll probably have questions throughout the course as they actually have to use it. You may just want to do some kind of quick demonstration in class doing a basic commit from scratch so they can ask questions about particular points.  I promise my mentor guide notes will get shorter from here. |
| 3 | Start with the normal stand up. What are you working on, what have you had trouble with, any project ideas so far. Tell them that they have plenty of time before the projects are due, but they can start thinking of things very roughly in terms of what they want their project to be about.  Students have found the videos this week pretty helpful so they may not have too many questions about the basics. If they have any questions about loops, lists, and all the other basics, answer questions about those. You can do some ad hoc demonstrations here to answer questions.  I included a guide about using Jupyter Notebook so they can see the ins and outs of the basics of using it. Of course, they can use any other kind of IDE they want but this is a nice guide for them to look through.  [Knowledge Check 1](https://colab.research.google.com/drive/1oRdniVHLdRQcg1sgHN68jVmWEUad2sER?usp=sharing)  We wanted to try something a little different, with knowledge checks in 3 week intervals to check basic understanding as well as to make sure they could push something to GitHub. These are pretty basic and shouldn’t take much time to verify they did them. I have instructions in the guide that they should make a repo and send it to you so you can keep track of their uploads. We look forward to feedback on this, but hopefully they find it useful.  Ideally you’d spend about 30 minutes per class going through the knowledge check, the goal is that it wouldn’t take up the entire class. |
| 4 | Again, start with standup! From here on out it will be more standardized and will involve a lot of helping them workshop project ideas and explaining concepts to them. I personally found loops very confusing the first few times I looked at them, so I included another YouTube video from a guy I like on there. Functions can also be confusing so included a video on that, as well as a Pluralsight video for importing data. |
| 5 | Similar to week 4, I found classes and modules kind of confusing the first several times I looked at them, so I included a couple videos I like that explain them. During stand up, encourage them to think of project ideas and tell them they should have a pretty good idea of what they want to do. If not, try to guide them towards a topic. |
| 6 | I included matplotlib’s tutorials on their website as well as a pluralsight video on plotting, so they may have different questions about building visualizations. This isn’t necessarily my strong suit, so I usually just reference the matplotlib gallery of plots if I need something complicated.  Students may start to drop the class at this point if they realize they don’t actually like Python or they get too busy with other obligations, so don’t worry too much if that happens. Reach out to them if so and see if there’s anything you can do, but it’s usually due to work conflicts and things like that. I think at least over Zoom the completion rate of classes has been around 60% of the starting number of students.  [Knowledge Check 2](https://colab.research.google.com/drive/1W0xUmX-pIP0ik99-jXzX3KJh8DIcqipu?usp=sharing)  Spend about 30 minutes walking through the knowledge check. |
| 7 | Have them choose a topic, even if they don’t necessarily feel inspired by it. The most important thing here is just making sure they’re working on *something* even if it’s just hacking away at a data set. You can jump into breakout rooms with students one on one and try to brainstorm ideas for them. At this point you can probably just pick a project idea for them if they’re still indecisive, which can work too. A few times I’ve just said “okay, you’re going to do your project on basic Natural Language Processing” which can work if they really just can’t choose between several ideas. A good fall back can always be “pick a random data set on Data.gov and have at it.” |
| 8 | Unit testing may or may not be involved in their projects at all, but it’s a good topic to go over in the video assigned this week, so we left it in there. You may want to do a quick demonstration or not depending on whether or not they have questions.  At this point, it will mostly be helping them troubleshoot their projects and come up with ideas for what direction they want to go with their data analysis. Encourage them to come to class with questions.  It can also be *hugely* helpful to other students to have one or two students that have started working on their project to show their code to the class, even if it doesn’t work. It’s actually maybe even better if it doesn’t work so the mentor can help them troubleshoot and explain to them how you would go about fixing their issue. It’s always very helpful for students as well to see other people at a similar point in their projects so they don’t feel like they’ve fallen behind. |
| 9 | Emphasize how you will download and run their projects. Students have gotten confused about how someone else would download and run a jupyter notebook (fair) so just explain that you’ll very literally just clone their repo and run it from the start. Tell them to restart the kernel and run all cells before uploading their notebook to make sure everything runs in order.  If they have an API key, students have used config files in the past and just DM’d their mentor the API key on slack to replace in the file. Anything works, as long as they aren’t uploading the API key to GitHub  [Knowledge Check 3](https://colab.research.google.com/drive/1DnJ_R321BIzlWyb8JocKhZNhfwzZ79vA?usp=sharing)  Spend about 30 minutes going over the knowledge check. |
| 10 | I cannot emphasize this enough: have the students send you their project or put it in the main slack channel so other people can try to run it.  One of the more easily fixed problems is a student not realizing their code doesn’t run on someone else’s machine until the night before the project is due (because of a path error in a file, for example, or because they forgot to tell you to install a certain package in the README.) This is essentially the last 3 weeks of class. Some people will have finished their project by week 6 and some people will be scrambling to finish at the end, which is fine! Just encourage them to have someone else as well as a mentor run their code on Windows, OS, and Linux (if possible.) Usually Windows and OS will be sufficient but it’s not something that might be obvious to them. |
| 11 | In the past, if students don’t have many questions about their projects at this point, sometimes we’ll go over resumes and job hunting tips. You, the mentors, have valuable perspective on that for them even if they’ve already talked to code lou staff, so they usually benefit a lot from interview tips / resume help / things of that nature.  Otherwise, it’s more of running their projects on your machine to see if you can get them to run. Encourage them to post their projects in the class slack channel so other students can see what they’re working on and also try to run their projects. Remind them to have their tech sessions / tech interview / project turned in at this point.  It’s usually useful as well to go through a project and grade it during class to show them how you will do that once you’re actually grading them. |
| 12 | Last class, woohoo! Remind them to have their projects turned in and help with any last minute issues.  90% of students will be done at this point, and 10% will tell you they don’t actually know how git or Python work and you’ll have to scramble to help them finish, hence my emphasis on running their projects in week 10 (or earlier). Regardless, just do your best to help them however you can and try to get them across the finish line.  Congratulations on finishing Data 1! |