**Github Instuctions Week 5**

This week we will, like last week, merge in the new lectures from the ‘upstream repository’ (John’s) into our local repository. The steps will be approximately the same as last week, but hopefully faster this time.

**0) Useful commands**

**git status** at any point will let you know the status of your repository. This should be like every other command.

If you find yourself in a mode where you can’t type, you are probably: (1) in a text editing mode, possibly from commiting changes without a message or using the **vim** command, or (2) currently using the **git log** command. In any of these cases, typing **q** or **ESC** + **:q!** Should allow you to resume controlling the command window.

If code or some command is taking to long to run or doesn’t seem to be finishing, you can always type: **“CTRL + C”** in the command window (running notebook or python) to kill it. You may need to press it twice in a row.

**1) Pregame (get our own changes/ repository in good shape)**

**Add, commit, and push** any changes you have made to your files in your repository. For a refresher on how to do this, refer to the document from [week 2 (STEPS 5-6 Only).](https://docs.google.com/document/d/1wiJDA2d23IrSuDUYf2UJbbDS6WJInsHT9Oy3ndn_AsY/edit?usp=sharing) Even if you mess up the merge, you can always revert to the repository at this point once it has been pushed.

**2) Party (In terminal/git bash)**

**git fetch upstream** (this downloads the content from John’s repo)

**git merge upstream/master origin** (this attempts to combine changes to both repos)

**3) Hangover (Deal with conflicts)**

There are qualitatively a few types of conflicts you will encounter, I will layout how to deal with those from easiest to hardest. Throughout the process… use **git status** to see which files still need to be dealt with.

1. Text based issues ie. README.md
   1. Edit file in favorite text editor… change to whatever you want.
      1. Fastest is **vim file\_name** if you are familiar with in line editors
      2. Just make sure to get rid of lines with <<<<<< or >>>>>
   2. **git add file\_name**
2. Annoying files we don’t care about or want eg. “.DS\_Store” files created by macs
   1. **git rm file\_name** (deletes and doesn’t track anymore)
3. Files that were added by John
   1. **git add new\_file**
4. Files which we both changed :(
   1. **git checkout --ours file\_name** (to keep your version… recommended)
   2. **git checkout --theirs file\_name** (to keep johns version… probably not what you want)

**4) Last resorts**

If everything gets messed up, you can use **git revert XX** where XX is the hash-key to a previous version (any time you committed some changes). The history of changes along with your commit messages can be had with **git log**, scroll with arrow keys, press **q** to leave this mode.

You can also find any old version the you pushed online on your github.com repository by clicking on XX Comits, selecting the commit you want, and selecting browse files. You can download any old version as a zip file.