**GITHUB INTRODUCTION – Day One**

It’s an exciting day, folks! We’re going to get you set up on GitHub. That way, you can put some of the programming projects you’ve done up on your GitHub website for the world to see. (One of the most important parts of a programming job or internship application is having some of your coding work visible on GitHub).   
In addition to making your code visible to prospective employers, here’s a scenario for why you’d want to use GitHub:

Let’s say you have a project you want to work on.   
Now you have **two primary motives:**  
**(a)**To write some code  
 **(b)**To take up the project as a team, meaning that more than one person is going to be responsible for writing the code.

Suppose you write a "Hello World!" program and then your friend/other developer decides that it should say "Hello, I am awesome" instead of the original "Hello World!".

Even for this little piece of code, you will have a copy on your computer and your friend will have one on his computer. Let’s say you change the code; now at this point your friend wouldn't know the changes you have made and vice versa.

This is where the Git magic comes.   
**1.**Git lets you save your code online.  
**2.**Git will allow all the developers of a project to see what changes the other one made

**The “Git” in GitHub**

To understand GitHub, you must first have an understanding of Git. Git is an open-source version control system that was started by Linus Trovalds—the same person who created Linux. Git is similar to other version control systems— Subversion, CVS, and Mercurial to name a few.

So, Git is a version control system, but what does that mean? When developers create something (an app, for example), they make constant changes to the code, releasing new versions up to and after the first official (non-beta) release.

Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

**The “Hub” in GitHub**

We’ve established that Git is a version control system, similar but better than the many alternatives available. So, what makes GitHub so special? Git is a command-line tool, but the center around which all things involving Git revolve is the hub—GitHub.com—where developers store their projects and network with like-minded people.

**I. GETTING STARTED WITH GITHUB**

**(Note: if you have Windows, you won’t follow the directions below. Follow the directions in this link instead:**

<https://www.pluralsight.com/guides/using-git-and-github-on-windows>

**Anything that needs to get typed into a terminal can be typed in by opening Programs – Anaconda – Anaconda Prompt).**

**For Mac users:**

**1. Install Homebrew**

Homebrew is a package manager. Once you have Homebrew installed, you can install other programs from the terminal, like Git.

A. Go to Applications – Utilities – Terminal to open up a terminal.

B. To install Homebrew, go to <https://brew.sh/>

C. There should be a command prompt like the one below that the website tells you to copy and paste directly into your terminal. Do so and press enter.

“/usr/bin/ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install)>"

D. It will tell you to press RETURN to continue.

E. Then, you’ll need to enter your system’s password (the password that you use to log into your computer when you turn it on)

F. You should receive an “installation successful message” in your terminal when the download is complete. This will take a few minutes.

G. Update Homebrew by typing “brew update” into your terminal and pressing enter.

**2. Install Git**

Now we can use Homebrew to install git. Just type “brew install git” into your terminal.

**3**. **Setup a GitHub email account.**

Go to <https://github.com/> and sign up for GitHub. You’ll want to use an email address that you’ll have even after you graduate from OES, because you’ll probably be using this for a while, maybe a lifetime! You only need the free version and you don’t need to check any of the extra boxes.

**4. Start a GitHub project on the GitHub website**

Click on “Start a project” or the “+ sign – New Repository” in the top right hand corner of the page. Either one will bring you to the same page.

Let’s create a Repository name called “Comment\_Project” and write in the description something like “Mimics OES Teacher Comments.” Keep the project public and don’t click on the README box. Then click on “Create repository”.

The webpage that you’ll be brought to will give you instructions for creating a new repo. Keep this webpage open but don’t follow the instructions for now. Instead, do this next…

**5. Create a reposition of the same name on your local machine**

Okay, now on your personal computer, create a folder called “Comment\_Project” inside your Unit8Student folder (note that this should be the same name as the one you set up in the last step on GitHub).

Copy ALL of the Python files you used for your Comment project into this folder.

In your terminal, type “cd Documents/Unit8Student/Comment\_Project” and press enter

In your terminal, type “pwd” and make sure it says “Documents/Unit8Student/Comment\_Project”

**6. Create a README file**

It is best practice to always include a README.md file in your repo. This is a file that explains what your code is and how it works. Note that .md stands for “markdown”. Markdown is a way to style text on the web. If you want to read more about it (it’s optional) you can go to:

https://guides.github.com/features/mastering-markdown/

**In your terminal, type:**

1. touch README.md *(then press enter)*

2. vim README.md *(then press enter)*

3. Type “i” to get into the insert mode

4. Write your file description. Something like “This program uses mimics OES teacher comments.”

5. Hit escape and then type “:w” enter and then “:q” enter to save and exit vim.

**7. Synchronizing your local** Comment\_Project **folder with the GitHub web-based** Comment\_Project **folder**

Okay, these next five steps are the most important. TYPE THEM IN EXACTLY. If you make even the slightest error it can be very difficult unless you are a Git expert. **PROOFREAD EACH LINE 5 TIMES BEFORE YOU PRESS ENTER!**

First to check that you are in the right place: **type “pwd” and it should say “Comment\_Project”.** Then from there, type:

1. git init *(then press enter)*

2. git add –A *(then press enter* - note this says add ALL the files in this folder to GitHub)

3. git commit –m “first commit” *(then press enter -* note that you can put whatever you want inside the quotation marks, but it is standard on the first commit to just say “first commit”)

It should say something like “Your name and email address were configured automatically.”

Go to the GitHub website. It should say something like “Quick Setup: If you’ve done this before” with a long url. Copy and paste that url and stick it into the following place inside your terminal:

4. git remote add origin *copied\_url*

Note: if you DO make a mistake entering the url, you can fix it by typing:

git remote set –url origin *the correct url*

5. git push –u origin master *(then press enter)*

At this point, it might ask for the GitHub username and password that you already set up.

**8. Look at your fancy pants webpage**

Return to the GitHub webpage that you have open and refresh the page. You should now see your comment project files on this webpage. Everyone can see your wonderful work!

**9. Troubleshooting**

If your website didn’t update correctly, you may have accidentally typed “git init” outside of the Comment folder. If so, do this:

1. While you are still inside the wrong folder, type: rm –rf .git

2. Do the directions again in Step 7 again while you are in the CORRECT folder

Other errors? Copy and paste the error code and “github” into google to see what solutions come up.

**10. Making Updates**

Okay, let’s say that you wanted to make updates to your comment program code or readme file. You can do that and resubmit the changes. Use VIM to go back into your README.md file and change one or two words around and resave and quit. Let’s now update Git with these changes. To do so, type:

1. git add –A

2. git commit –m “updated readme”

3. git push –u origin master

**Homework**

Inside the Day3Git folder, there is another folder called gitpracticeNAME and a file within that called practiceNAME.py. Change both the folder and the file to include your name instead of NAME. For instance, your folder would get renamed to gitpracticeJOHN and your file would get renamed to practiceJOHN.py. Make a new repo on GitHub that has this same folder name (for example, gitpracticeJOHN) and follow the steps above again to synch your local repo with the GitHub repo. DO NOT fill in the functions contained inside the practice file. We will do this tomorrow with partners.