What is git:

- Distributed version control system. Or a system that records changes in our files and alows us to acces specifc versions of those files at a given time
- It also allows you to work collaboratively by allowing each person to work on a copy of a project before syncing them
- syncs with Github

Intalling Git:

- download from Git website: Git (git-scm.com)
    - It's also a refrence website
- For windows, you can also use the command line
- After installing you can check by opening your terminal and entering: *git --version*

Github:

- Most popular, and free to use remote repository
- Create a Github account
- allows you to host your repositories
- can also share and discover other user's repositories

Tip: create a folder to store all of your local repositories

For this tutorial we'll be using Gitkraken:

- Install it
- Once you have it installed, open it. You will see the option to Log In. Log in Using your Github account
- Next you will see a screen that asks "How do you want to get started?" with the following four options
    - Open repo
    - Create repo
    - Clone repo
    - Group repos
- Select: Create repo.
- At this point you can also select the Github.com option which should bring up the option to connect your Github with GitKraken
- For now create a "Local Only" Repo
- If the "How do you want to get started?" page didn't appear or if you want to create a repo once the you've alredy started using the program, you can click on File and select "Init Repo" or click *ctrl + I* on you keyboard

- Let's call this "Learn-Gitkraken" and select the folder you wish to initalize the repo in.
- For the default branch name, if it says main leave it alone, and if not give it the name *main*
- Click "Create Repository"
- If you look at your computer's library you should now be able to see your repo with a README file already created

Next we'll add a file to the repo

- In the repo folder add a new text file called *text_for_git.txt*
- In GitKraken you will now see on the right panel there was a file change in the working directory
- Click on "view change". This will bring up the staged and unstage files
- You will see that our text file is under the Unstaged Files section
- Click on the "Stage all Files Button"

What is Staging and Committing:

- In git we keep a history of commits
    - they're kind of like a save points for code that you can revert to at your choosing
    - Notably this works with an entire project, not just individual files
- The files that you want to be part of your commit first go on the git index which is "staging" them
- You can also put in a commit message, let's say "first commit"
- Then click "Commit Changes"
- You will then see on GitKraken's UI that there is a change in the commit history

Let's Try adding another File:

- let's call it second.text. It is automatically added to our unstaged file section
- Stage it and commit it
- Let's add a third file called "third" and stage and commit it.
- You GitKraken should now look like this

Revert Commit:

- Let's say we want to go back to a version of our folder without "second.txt"
- Right-click on "Second Commit" which will bring up a menu and select "Reverse commit"
- GitKraken will ask you to confirm if you want to revert. Select Yes

- This will remove the file from your folder and show a new commit in your history called "Revert Second Commit" as seen below

In addtion to Revert Commit you can also Rest Commit in order to manipulate your commit history, but what is the difference between them?

- Start be adding a new file called "fourth.txt" and stage and commit it
- Add a fifth file called "fifth.txt" and stage and commit it
- Your commit history should look like the below image…
- Right click on the second commit. In the menu click reset main to this commit and scroll down to "Hard - discard all changes"
- This will revert and remove all of the changes after "Second Commit" and it will bring back the "second.txt" file

Tagging

- You can also add tags to the Branch or a Commit by right-clicking it and and scrolling down
- They serve as a way of marking down an important part of your project and making it easly search able by look at the "Tags"

Branching:

- Branches give coders the abiltity to work in an isolated version of the code that allows them to work without affecting the overall project. If the work they do is good and works they can then merge their work into the orignal code or the "Main" branch. If they did a bad job, they can simply delete the branch with no negative effects
- To create a new branch, right click on the "main" branch and select "create branch here". Let's name this branch "secondary".
- Your GitKraken should now look something like this…
- Let's create two more bracnches called "tertiary" and "quaternary"
- Make sure you're in the main branch
- In your repositary's folder, create a new text filed called "hello_world" and type "Hello World" into it then save. Stage and commit this file

Switch to the "secondary" branch.

- You'll notice that if you look at your folder while on the secondary branch you cannot access the "hello_world" file. Create a new text file (let's call it hello_again) and fil it with some text, let's say "Hello my baby, Hello my Honey"
- Stage and commit the file. Your GitKraken should now look something like this…

- Let's add another text file to the Secondary Branch. Stage and Commit it. Gitkraken should look something like this…

Let's Merge secondary into main:

- Select/enter the main branch. right-click on secondary and select "Merge secondary into main".
- Your gitkraken should now look something like this…
- Now the text files from secondary AND main are in the main branch of the repository

Select the tertiary branch

- Create a text file and commit it
- You can also branch off a branch the same way you branched off of main
- righ-click the tertiary branch and create a new branch.
- I called this branch of tertiary "tertiary_sequal"
- add a text file to your new branch and commit it
- add and commit another text file
- Your gitkraken should look something like this…

Switch to the Quartinary Branch

- Right Click on your latest commit in "tertiary_sequal" and select "Cherry pick commit"
- This allows you to only merge changes from a single commit rather than an entire branch
- Your git kraken should now look something like this…

Now let's merge everything into main

- Your GitKraken should now look something like this…
- You can also delete branches by righ-clicking and selecting "Delete 'Branch Name' "

Merging Conflicts

- Happens when more than one person works on the same file on different branches and try to merge them to the same branch
- Best way to fix it is to not have it by having good communication, but let's see how you fix it when it occurs
- Let's say you have a merge conflict with a text file. Open the file, it will show the differences between both files and you'll have to edit it manually to see what you want to actually keep and commit.

- Now commit and merge as normal

Let's look at cloning Github Repositories:

- Click on File or the File Icon and select Clone or Clone repo
- Here you have the option to Clone based off a URL or, if you select the Github.com tabe, you have the option to clone a repository from your Github account
- For the purpose of this tutorial, I created an empty repository on Github.com called learn-git-continued. I left it public and clicked the box to add a README file
- Now clone the repo you want from Github
- Now let's add text file into the cloned repo in the folder you cloned the repo into. Let's call it "hello_again.txt". Stage and commit it.
- If you look at your repo in Github, you will notice that it doesn't have the commit we just made

To get the change from your local machine to Github, you have to Push it to your Github.

- Click the Push button
- If you look at your Github you will see that the it now has the file. If you don't see it, refresh your page.

Let's Try adding a file from our Github to our local machine

- Let's call it git_hello and commit changes
- To sync this change to your local machine, click on the Pull Button
- You should now see that the file from your Github (your remote repo) has been added to your local machine on GitKraken

Pull Requests:

- To help manage the project people will use something called Pull Request to get files from the remote repository. Essentially you're asking permission to access the files
- Add a new file to your remote repo, let's call it "pull_practice"
- To make a Pull Request, hover over the Pull Request menu and click on the green plus button
- The following menu will be brought up. Select the repo you want to pull from and where you want it as well as the branch you want to pull from and commit to.
- Finally add a title and click "create pull request"
- You will then see on Github that you have a recieved a notification for a Pull Request and can choose to allow it or deny it