git

Version control software

Tracks changes to files within a directory

Allows multiple people to collaborate on a project

Tracking Changes without VC

+ Main_File.txt   2.1 kb
+ Main_File_old_1.txt   1.0 kb
+ Main_File_old_2.txt   1.2 kb
+ Main_File_old_3.txt   1.3 kb
+ Main_File_old_4.txt   1.6 kb
+ Main_File_old_5.txt   1.8 kb

- One current version of the file
- Multiple copies with different changes in each copy
- Each new file will probably be bigger
  - This wastes space
- What happens when two people are working on the same document?
  - What happens if they both make a change to the same sentence?
  - What happens if one person deletes a sentence?

What Git Does

Version1

```
1|  results = []
2|  for num in range(20):
3|      if num %2 == 0:
4|          results.append(num)
5|
6|  print(results)
```

Nothing has been saved
yet so the entire file is
saved

Version2

```
1|  results = []
2|  for num in range(20):
3|      if num %2 == 1:
4|          results.append(num)
5|
6|  print(results)
```

All that changed is one line, so all
that needs to be stored is:

- if num%2 == 0:
+ if num%2 == 1:

Git Workflow

1. Open the terminal (Osx/unix) or Git Bash(Windows)

2. Navigate to the directory you would like to track
    a. $ cd path/to/directory

3. Initialize git to track the directory
    a. $ git init

4. Add files/changes to be tracked
    a. $ git add .
        i. This will add all the files within the directory
    b. $ git add 'filename'
        i. This will add specific files

5. Commit the changes, this saves a backup for how the files currently look
    a. $ git commit -m 'commit message'
        i. -m is a flag for message, the text in quotes after the flag is the message

Useful Git Commands

$ git status
-	tells you which branch you are on
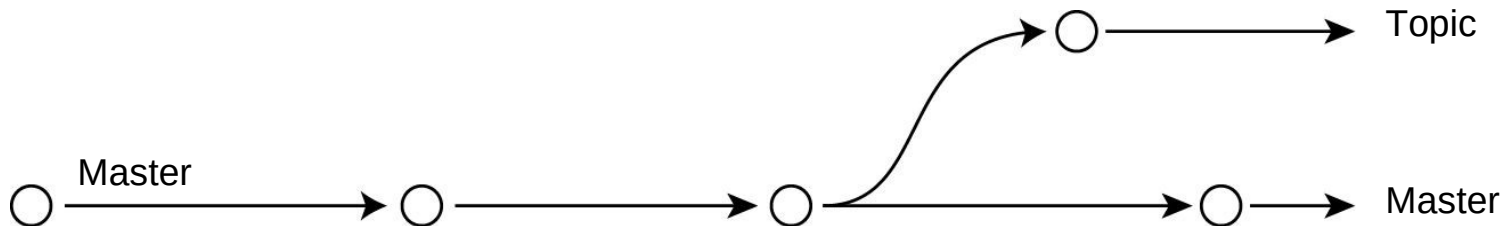-	tells you which files have been modified

$ git diff 'filename'
-	shows you what changes have been made to the file since the last commit

$ git reset 'filename' or '.'
-	opposite of $ git add 'filename' or '.'

Branching

$ git branch 'branch_name'
-        This will create a new branch
$ git checkout 'branch_name'
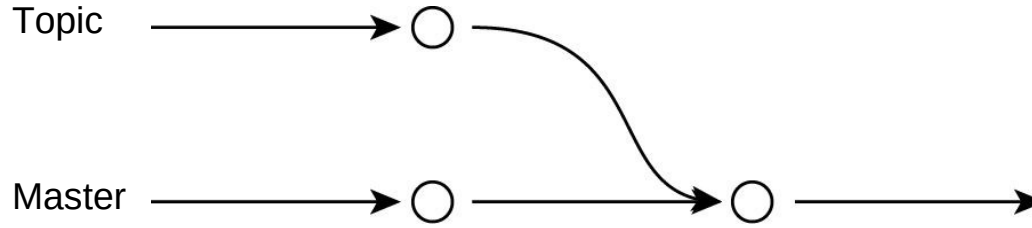-   This will change which branch is being used

$ git branch
-        this command provides a list of branches
-        an * is next to the active branch

$ git status
-        This will also tell you which branch is active

Merging

Topic ⟶ ◯
Master ⟶ ◯ ⟶ ◯ ⟶

$ git checkout master
- you always merge the changes into a branch
$ git merge Topic
- this will merge topic into master

Can result in merge conflicts

github.com

Allows you to store your code online
- This makes it easier to share code with collaborators
- It also is a more robust method of backing up your code

1. Create a new repository on github.com
2. Link your local repository with the online one
   a. $ git remote add origin https://github.com/user/repo.git

3. Check that the link is correct
   a. $ git remote -v

4. Push changes to github
   a. $ git push remotename branchname
   b. $ git push origin master

5. Pull changes from github
   a. $ git pull remotename branchname
   b. $ git pull origin master

Most of gits functionality is available in atom
- Repositories must be initialized and linked to the remote from the terminal first

More guides on git
https://guides.github.com/
https://www.atlassian.com/git/tutorials
https://git-scm.com/doc