```
### 16. Instructor Do: Intro to Git (0:30)
###
* Explain to students that so far GitHub has really only been used as a sort of
drop box to store our files. Although GitHub works well this way, it has far
greater capability. Today there will be a deeper dive into what Git is and how
to use it through the terminal to interact with Github.

* **N.b.**: If teaching with VS Code, consider using the [Git
History](https://marketplace.visualstudio.com/items?itemName=donjayamanne.
githistory) extension to illustrate this section's concents.

![Visualizing Git histories with the Git History
plugin](https://raw.githubusercontent.com/DonJayamanne/gitHistoryVSCode/master/
images/gitLogv2.gif)

* Open [Intro_to_Git](Resources/Intro_to_Git.pptx) to go over slides 1-22.
Explain that Git is essentially a way for us to keep track of our work over
time.

  * Explain that, whenever we get another piece of a project working, we can
  save the change with Git.

  * Explain that this "save" is called a **commit**, and represents a
  "checkpoint" for our project.

![A commit is a lot like a changelog
note](https://cdn-images-1.medium.com/max/1600/1*zj-d8TopjgBml2QVM-672w.jpeg)

* Explain that, if we break something in our code while developing, this system
allows us to restore the working code from before.

* Explain that, since Git remembers these "checkpoints", we can work on several
different concerns all at once.

  * Suppose we need to analyze Uber ride data for our project.

  * Explain that we might decide to analyze the average age of riders. Git
  essentially allows us to write this code, and save it with the name: `age
  analysis`.

* Emphasize that this code is _different_ from the code we started with, and
that it lives separately from it.

  * Explain that, in this scenario, we have a version of the code, called
  `master`, which is the "main" version of our code; and a version, called `age
  analysis`, which contains updates.

* Explain that each version of the code lives on a different **branch**.

  * Explain that a **branch** is essentially a history of changes.

  * Explain that, in this case, we say that the `age analysis` branch
  **diverged** from the `master` branch.

  * Take a moment to demonstrate the difference between the files on the
  `age_analysis` and `master` branches.

* Explain that saving the age analysis code in a different branch gives our
teammates a chance to review it for errors and offer suggestions.

* After the proposed change has been reviewed, we can update `master` branch to
include the changes in `age analysis` by doing a **merge**.

* Explain that **merging** two branches turns them into one.

* Explain that this is how we can work on new features or bugfixes without
making changes to code we know is working.

  * Explain that this also makes easy to work with teammates, as people can
  avoid stepping on each others' toes by working on different branches.

* Finally, take a moment to review Git's "Snapshot model":

> "...Git thinks of its data more like a set of snapshots of a miniature
> filesystem. Every time you commit, or save the state of your project in Git,
> it basically takes a picture of what all your files look like at that moment
> and stores a reference to that snapshot. To be efficient, if files have not
> changed, Git doesn’t store the file again, just a link to the previous
> identical file it has already stored. Git thinks about its data more like a
> stream of snapshots."
>
![Git Snapshot Model](https://git-scm.com/book/en/v2/images/snapshots.png)

### 17. Everyone Do: Adding Files from the Command Line (0:10)
###
* Tell students that so far they have only added files using the GitHub website,
which works well when just dealing with one or two files. What happens when
multiple files need to be quickly added?

  * The command line comes to the rescue!

* Have students follow along with creating a repo and adding files with
Terminal/Git-Bash.

  * Create a new repo.

  * From repo page, click the green box in the top right "Clone or download",
  select "Use SSH" and copy the link to the clipboard.

  ![clone repo](Images/GitClone.gif)

  * Open terminal (or git-bash for Windows users) and navigate to the home
  folder using `cd ~`.

  * Type in `git clone <repository link>` in the terminal to clone the repo to
  the current directory. Once this has run, everyone should now see a folder
  with the same name as the repo.

    ![terminal clone](Images/GitClone_command.png)

```

~~~
  * Open the folder in VS Code and create two python script files named
  `script01.py` and `script02.py`.

  * Once the files have been created, open up Terminal/git-bash and navigate to
  the repo folder. Run the following lines and explain each as you go through
  them.

  ```bash
  # Displays that status of files in the folder
  git status

  # Adds all the files into a staging area
  git add .

  # Check that thr files were added correctly
  git status

  # Commits all the files to your repo and adds a message
  git commit -m <add commit message here>

  # Pushes the changes up to GitHub
  git push origin master ```

  * Finally navigate to the repo on [Github.com](https://github.com/) to see
  that the changes have been pushed up.

* Make sure every student was able to successfully clone a repo, add file to the
repo, commit the changes, and then push the changes to Github all from the
command line.

### 18. Students Do: Adding more to the repo (0:15)
###
* **Instructions**

  * Using the repo that just created, make or add the following changes:

    * Add new lines of code to one of the python files. * Create a new folder. *
    Add a file to the newly created folder. * Add, commit and push the changes.
    * Delete the new folder. * Add, commit and push the changes again.

### 19. Instructor Do: Review Git (0:10)
###
* Ask students for any questions students may have and take a few minutes to
review any commands which weren't clear. Offer to help students with this
throughout the day and during office hours.

* Explain to students that this will be the new, primary way of submitting
homework to GitHub (no more manual uploads!).

* Reassure them that it's ok if this take some time to figure out. By the end of
the course, they will be git ninjas!

* Encourage students to continue to add and commit their activities today into a
repo for additional practice.
~~~

```
### 20. Instructor Do: Video Guide and Close Class (0:02)
###
* Before finishing up for the night, slack out the [Video
Guide](../VideoGuide.md) containing walkthroughs of this week's key activities.
Encourage students to review them later and utilize office hours if they have
further questions.

### Extra Do: Additional exercises
###
* If class finishes ahead of schedule let students know that there are some
additional challenging exercises to work for those that are ready. For students
that still have question there will be time to get additional help from TA's and
the instructor.

* Supplemental exercises:

  * [ADVANCED_Ins_Set_Operations](Extra_Content/ADVANCED_Ins_Set_Operations)

  * [ADVANCED_Stu_Resume_Analysis](Extra_Content/ADVANCED_Stu_Resume_Analysis)

  * [ADVANCED_Stu_UUID_Generator](Extra_Content/ADVANCED_Stu_UUID_Generator)

  * [Stu_Email](Extra_Content/Stu_Email)

- - -

### LessonPlan & Slideshow Instructor Feedback
###
* Please click the link which best represents your overall feeling regarding
today's class. It will link you to a form which allows you to submit additional
(optional) feedback.

* [:heart_eyes:
Great](https://www.surveygizmo.com/s3/4381674/DataViz-Instructor-Feedback?
section=python-day-3&lp_useful=great)

* [:grinning:
Like](https://www.surveygizmo.com/s3/4381674/DataViz-Instructor-Feedback?section
=python-day-3&lp_useful=like)

* [:neutral_face:
Neutral](https://www.surveygizmo.com/s3/4381674/DataViz-Instructor-Feedback?
section=python-day-3&lp_useful=neutral)

* [:confounded:
Dislike](https://www.surveygizmo.com/s3/4381674/DataViz-Instructor-Feedback?
section=python-day-3&lp_useful=dislike)

* [:triumph: Not
Great](https://www.surveygizmo.com/s3/4381674/DataViz-Instructor-Feedback?
section=python-day-3&lp_useful=not%great)
```