Recipe for Success

AKA How to Survive and Thrive at the Flatiron School

GENERAL BEST PRACTICES

- **Life happens, try to stay one step ahead**
- When something happens, **talk to us!** If you communicate what's going on, we can work with you - let us help!
- Having a **dedicated space to study** - a study corner, a coffee shop or library, a desk in a quiet workspace, or whatever can be consistent for you - can help you get focused and stay on task!
- Organizing unofficial peer-to-peer lab review sessions or other impromptu study groups to move through the material with others working on the same part of the curriculum is **highly recommended** (let us know if you need a zoom room!)
- This is your time and your future (and your money!) - **jump in with both feet**
- **Ask questions frequently**, not only of your instructor but also of other students
- **Help your fellow students out when they ask questions** (explaining it activates different/stronger memory circuitry in your brain)
- **Reach out for help as soon you begin to slip, rather than waiting**
- Talk about your feelings **(you're definitely not the only one feeling that way)**

GRADED ASSIGNMENT PROTOCOL

Our expectations for Checkpoints and Code Challenges:

- Assessments are **open book** (ok to reference labs, lessons, and Google)
- Be on **zoom** and in a breakout room
- **No copy and pasting code**
- **No screen sharing or messaging** with peers during assessments
- **The use of generative AI such as Chatgpt is prohibited**
- Exit break out room after submission

PLAGIARISM

Plagiarism is grounds for dismissal from the program - but what exactly is plagiarism when it comes to writing code?

A quick definition of plagiarism, from the Student Catalog:

> "Plagiarism is defined as the use of work or concepts contributed by other individuals without proper attribution or citation. Unique ideas or materials taken from another source for either written or oral use must be fully acknowledged in academic work to be graded. While students are encouraged to use outside

> resources to source and build their projects, the cloning and copying of repositories or other resources is expressly prohibited."

The key is how you present the work: are you misrepresenting work that is not yours as if it was written by you? If so, we may have a plagiarism issue.

The easiest way to avoid this is to make sure you explicitly **both link the original content and explain how you used or adapted that content**. This best practice both provides attribution, by directly linking to where you found the content, and provides context, showcasing your own understanding of the code and why it was useful to you.

Remember - you're trying to build projects in such a way that your own skills are showcased. Googling and learning from others are a vital part of programming, but misrepresenting your process or what you are capable of doing will only hurt you.

Please refer to the Student Catalog for the full details around our official policies, including a description of the potential disciplinary actions which may be taken if you are found to be in violation of our plagiarism policy.

FEEDBACK

We ask that you follow the CASK protocol when giving feedback to others!

- **CONSENSUAL** - If someone isn't in a good space to receive feedback, it won't stick
- **ACTIONABLE** - Outline ways to change or act on the feedback, instead of saying something vague like "this is good" or "this is bad"
- **SPECIFIC** - Give examples when you can, to anchor your feedback in a real way
- **KIND** - The goal is to help someone improve, not to belittle others

This is also a good place to reference our Flatiron School Community Guidelines.

COMMUNITY EVENTS

Community events are exactly that - events designed to connect you with your peers and support you during your Flatiron School journey. Some of these events are Data Science-specific, while others are open to everyone across all the disciplines taught at the Flatiron School.

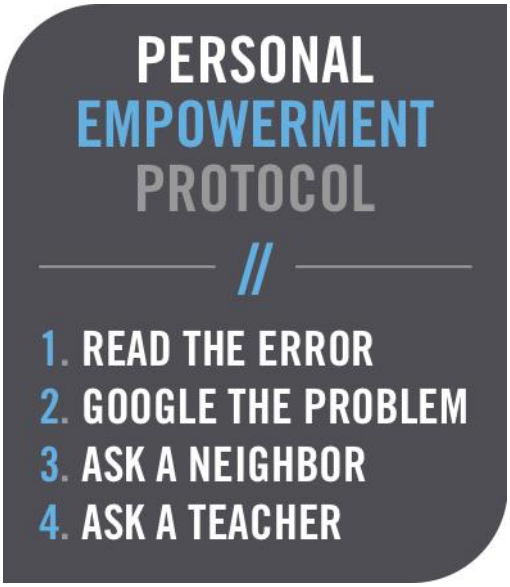
Flatiron School Community Events Calendar

See also: Canvas Homeroom

CANVAS & CURRICULUM

General Curriculum Guidelines

- It is often easier to read through the materials, then go back and do a topic's labs **after** reading the content, focusing specifically on concepts that seem less clear
    - Often, concepts introduced later in the topic will help you complete the labs earlier on
    - It is 100% okay to have the lesson open in one window while doing the lab in another
- When doing the labs, follow the **Personal Empowerment Protocol** when you get stuck:
    - Read the errors you see, focusing on where the error is popping up and what it recommends at the bottom
    - Then, Google it! Either type out what you're trying to do, or copy/paste the error message
    - If you're still stuck, check the slack channel for that phase to see if anyone else has run into the same problem
    - **Advice:** set a timer, and give yourself at least 10 minutes to puzzle through something difficult that you're encountering before directly asking others (classmates or instructor) for help



- It is **totally okay** if you get stuck and want to explore the **solution** of a lab to get past a tricky session!
    - But you should still go through and type out the code yourself (muscle memory is helpful and will help the code sink in). **Even better:** rewrite the solution in a way that seems more intuitive to you, or practice turning repetitive code into functions.

Checkpoint & Code Challenge Protocol

1. At the time set out on the calendar, join the cohort zoom and I'll put you into individual breakout rooms (stay there until you're finished!)
2. Assessment notebook will be provided via canvas
    a. We will cover this in depth the first few days so don't worry
3. Do the work! (and remember - follow the Graded Assignment Protocol!)
4. Save as you go! Save, save, save
5. When completed/finished the assessment needs to be submitted via canvas and codegrade

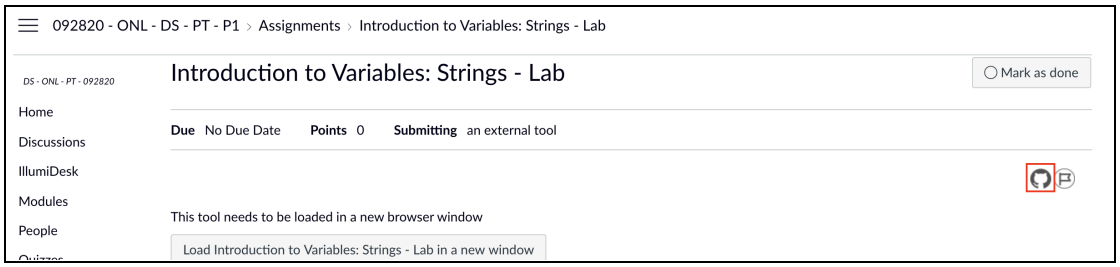
6. Click into the appropriate Canvas Phase course and navigate to the appropriate assessment
7. Load assessment in CodeGrade (will open a new browser window) and submit the notebook with your code

Once the checkpoint or code challenge has been graded, you can fetch your feedback by returning to the Canvas assignment page and loading up CodeGrade again.

1. Your grades and originally submitted notebook can be viewed with instructor comment cells
2. Final grades will be transferred to canvas gradebook and can also be viewed there

Use Git/GitHub With the Curriculum Content

- If you ever want to work on a lab **locally** instead of in SaturnCloud, you can access the GitHub repository for each of the labs by clicking on the top GitHub icon on the page

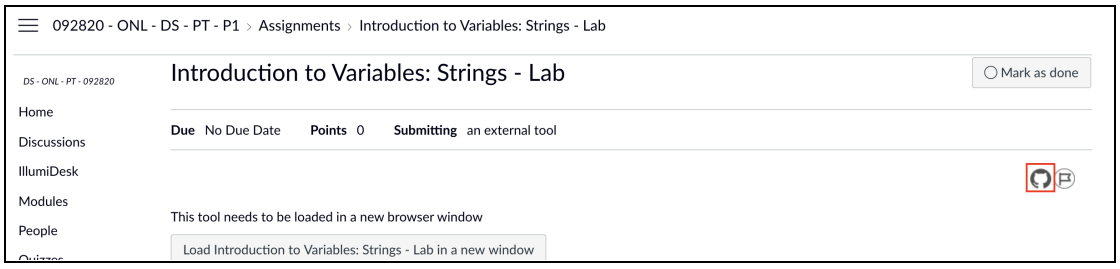


  - From there, create a fork of the lab, then clone it onto your local computer. When you're done with the lab, you can then add/commit/push your changes and code back up to your fork on your own GitHub
  - Remember the golden rule of git: **Never clone a repository into an existing repository**
- If you ever want to look at a jupyter notebook online, but it won't render on GitHub, you can always use the **notebook viewer tool** to view any notebook that has a URL
  - For example, this notebook won't render on GitHub, showing this error:

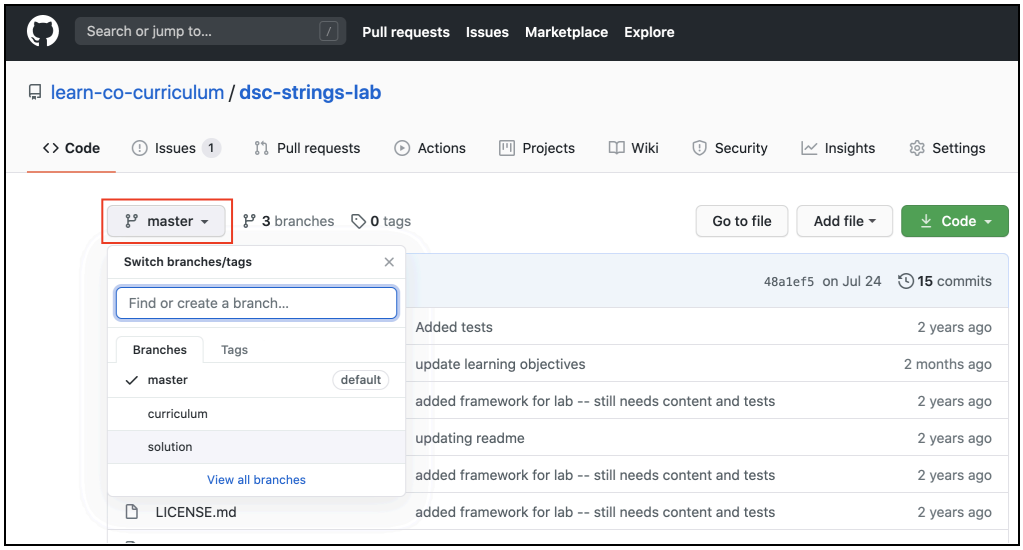
- But I can paste the whole URL for the notebook and render it using NBViewer:

Check the Solution of a Lab

1. Click the GitHub icon in the top corner of the lab on the Canvas page



2. Change the branch you're viewing from master/main to solution



3. You can scroll down to view the README of the solution branch, which will showcase the solution code
4. If you're working on a lab locally, and would like to **access the solution branch on your local computer** instead of loading it on GitHub, you can:
    a. Make sure you're in the cloned repository for the lab (either your fork or the original repo)

b. Run `git checkout solution` (**without -b!**) to checkout the solution branch, tracked from the remote branch

```
(learn-env) Lindsey-Berlins-MB-Pro:dsc-implementing-statistics-with-functions-lab lberlin$ git checkout solution
Branch 'solution' set up to track remote branch 'solution' from 'origin'.
Switched to a new branch 'solution'
(learn-env) Lindsey-Berlins-MB-Pro:dsc-implementing-statistics-with-functions-lab lberlin$
```

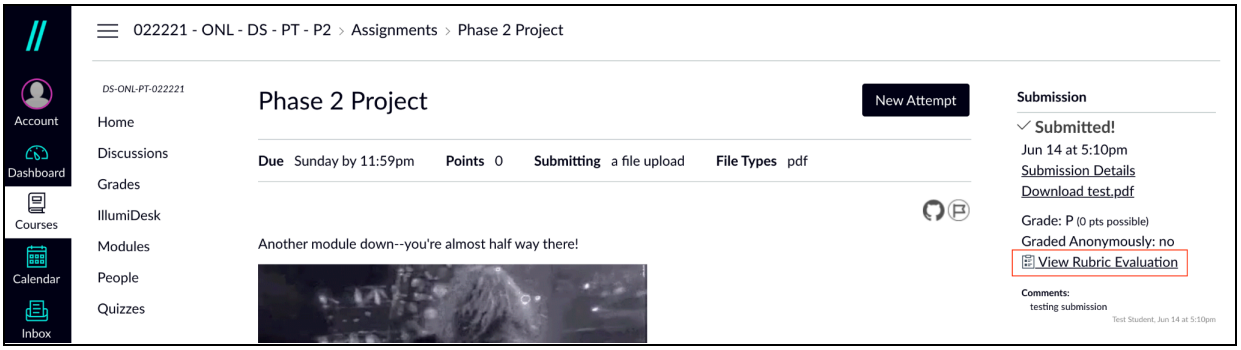
c. Then, if you re-open the jupyter notebook from that branch, it will have the solution code loaded

Please note: **only labs** have a solution branch (code lessons do not)

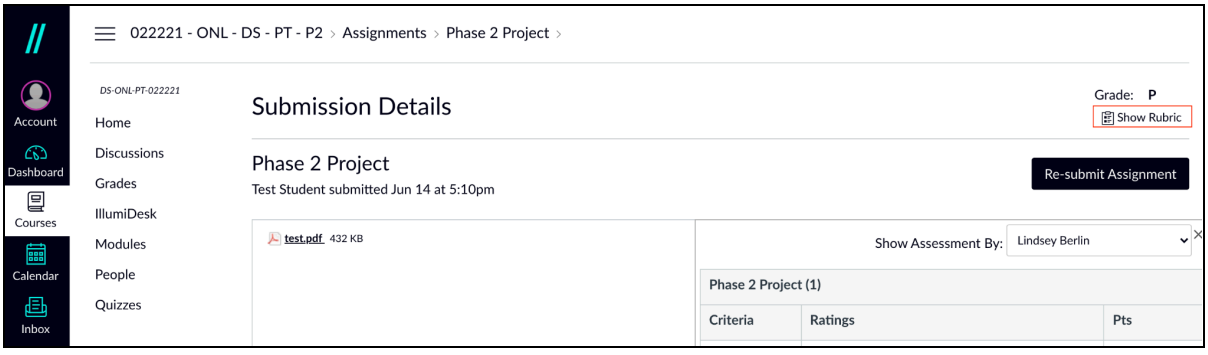
View Project Feedback

After a grade has been input in Canvas, you are able to see all of the written feedback on the rubric connected to that assignment.

From the project assignment page in Canvas, click “View Rubric Evaluation” to navigate to the feedback page.



From the Submission Details page, click “Show Rubric” if you cannot see any written feedback or the grading on the rubric.



Give Feedback on the Curriculum

- Flatiron School has embraced an open-source curriculum to reflect industry trends and feedback - practice both contributing to an open-source community and articulating and documenting your findings by sending in your feedback often!
- **Our Curriculum team solicits your input on lessons and labs in two ways:**

1. At the bottom of each lesson, pages ask “How do you feel about this lesson?” Share a quick “Thumbs Up” or “Thumbs Down” about the given lesson or lab.
2. Underneath the Thumbs Up/Thumbs Down feedback mechanism, it asks for specific feedback. This, or the “Flag” icon at the top of the lesson, will take you to the GitHub Issues page for that lesson or lab where you can write out what’s broken - especially useful for any issues with the code that you find!

BLOGS & NON-TECHNICAL WEB PRESENCE

You are encouraged to write, publish and submit **at least four blog posts** during the course of the program - **due early in the second week** of Phases 1 - 4.

- Why do we make writing blog posts a recommendation of the program? **Because it gets people jobs.**
- As covered in the Blogging Overview page on Canvas, blogging has many benefits:
    - **Develop your written communication skills.** Your writing ability will be critical to your success when completing job applications and presenting your work to colleagues. Blogging is great practice for identifying and clearly communicating the most important points of any subject.
    - **Demonstrate your talent to employers.** Potential employers will review your blog to determine whether to offer you an interview or a job. Some students have even been invited to interview or exempted from technical interviews based on their blogs.
    - **Strengthen your knowledge.** Blogging helps you explore new topics, deepen your understanding, and crystallize what you've learned.
    - **Help your peers and the broader community.** Have you ever Googled a question you had and found the answer on a blog? Writing blog posts helps others who are following in your footsteps!
- Please refer to the content on Canvas for more details about your blog, including suggestions for where to write (what platforms to use)

Blog Requirements

Each blog must:

- Be composed primarily of original material written by you
- Include proper attribution for contents not written/created by you
- Have high-quality content and formatting (use Grammarly!)
- Needs to be online (working URL directing to that post)
- Around 1000 - 3000 words in length

Blog Topics

- **Phase 1:** Why did you decide to learn data science?
- **Phase 2:** Pick a data science tool / library that you would like to learn more about, and summarize it.

- **Phase 3:** Write a tutorial (with a data set and code sample) on something that you think might be interesting to other people taking the course.
- **Phase 4:** Pick a research paper / case study about data science and write a blog post to explain it to a non-technical business stakeholder.

Want to write about something else? Feel free to propose a new topic to your instructor!

Cultivate a Professional Data Science Presence Online

A blog is one way to cultivate your professional online presence. Here are a few more ideas:

- Create a GitHub profile README, which can work as a personal website
- Start marketing your LinkedIn presence early on, share projects, blogs and helpful posts
- Create an actual personal website
- Create a YouTube channel, where you share recordings of your non-technical project presentations (as well as any additional content you may want to create!)
- Use Tableau or other dashboarding to create a profile/resume
- Websites like Datacamp and Kaggle allow you to create profiles as well
- If you have other ideas, **please share them** on Slack!

ZOOM

- **Please keep your video on** during sessions (it is impossible to talk to a room of blank screens, and many of these sessions are designed to be interactive!)
- **Please mute your mic** when not asking a question/talking
- When pair programming, you can use one person's computer and **use the Request Control feature** to swap roles
- **Create your own Zoom rooms to meet with your cohort mates**
    - You can use a Zoom room for **pair-programming** beyond the structured pair programming sessions
    - It's just as useful for answering questions
        - It's often easier to talk through difficult questions/answers than to type them out
        - If you're comfortable, paste the link in the cohort Slack channel so anyone interested can attend
    - To create your own zoom room: open Zoom, click New Meeting, which launches a room
        - Click the Invite button on the toolbar
        - The easiest share method is to click "Copy Url" and paste into Slack
    - You can always ask a staff member if you'd like a room created for you (which won't have the 40 minute limit of the normal free tier!)

- When in breakout rooms, note that the Chat feature only shares messages with your own breakout room members - if you'd like to ask for help or ask a question more broadly, please use Slack instead!

PAIR PROGRAMMING

- Pair programming is an agile programming practice where **two people work together on the same problem on the same computer at the same time**
    - This is not just a Flatiron School thing! This is a common practice in companies with agile workflows
- Benefits of pair programming: catch errors more quickly, share the workload, get immediate feedback, solve problems in different ways
- How to pair program: find a partner, and decide a problem to work on
    - Two roles:
        - **Navigator**: person directing the solution, leading the discussion on what the code should look like, and focusing on problem solving and design rather than syntax
        - **Driver**: person in control of the computer/keyboard actually typing, focusing on translating the Navigator's ideas into actual code (typically this is the person with less programming experience)
    - Join a Zoom room (see above) and share your screen, using remote control to switch roles.
- Labs make for great material to use for a pair programming session!
- **Best practices:**
    - If you're the **Driver**: ask a lot of questions! Make sure you understand every bit of code you type.
    - If you're the **Navigator**: be clear and patient, and over-communicate what you mean.
    - Switch roles after a while! Make sure everyone has the chance to both drive and navigate.
    - When giving feedback, follow the CASK rule: make your feedback Consensual, Actionable, Specific and Kind.
    - Remember! The point isn't to barrel through a lab or a problem, but to **write better code, solve a solution more elegantly, and learn from each other!**

TAKING NOTES

- Taking coding notes is a bit different from taking notes for other things you've learned, as you'll often want to copy/paste code snippets (which is hard if you take notes by hand!)
    - This isn't to say that handwritten notes aren't useful, they're just more useful for making notes about things like definitions than they are for code.
- My suggestion when taking code notes/making code snippets is to **make the code generalizable** when you copy it into a note. That means giving examples that you'll

understand in any context, rather than something really specific to the case when you saw the useful bit of code.
    - Example:

Quickly changing values in a column using a dictionary map and list comprehension

```
animals = {"cat": 0, "dog": 1, "rabbit": 2}
df["Animal"] = [animals[item] for item in df["Animal"]]
```

- **Where to take notes:** you can take notes on any note-taking app on your computer, but I personally enjoy Boostnote - it's free, open-source, and supports markdown syntax (same as what you use to edit README.md files)
    - Need help with markdown? Here's a cheat sheet!

MORE RESOURCES

- Sometimes you'll still have questions after reading through the curriculum, attending the Lectures, and going through the labs - **and that's okay!**
- Your instructor likely has a whole host of additional resources at their disposal, **please just ask** if you'd like an additional resource to engage with some aspect of the material in a different way
- You're encouraged to search for blogs or YouTube videos or other resources around the web to learn more about specific topics. Some suggestions:
    - Blogs/Sites:
        - Machine Learning Mastery
        - Chris Albon's Notes
    - Subreddits:
        - r/DataScience
        - r/DataIsBeautiful
    - Podcasts:
        - Data Skeptic
        - Towards Data Science
    - YouTube Playlists:
        - StatQuest
        - 3 Blue 1 Brown (for math concepts and neural networks)
    - Free Books:
        - Think Stats
    - Free Courses:
        - Kaggle's Data Science Courses
        - Khan Academy (both for math and for SQL)
        - Google's Python Class
- If you find a resource you like, or a specific post/podcast episode/video that's helpful, **please share it** in an appropriate Discord channel!