

About this Book

Contents

Syllabus

- [About](#)
- [Tools and Resources](#)
- [Data Science Achievements](#)
- [Grading](#)
- [Grading Policies](#)
- [Support](#)
- [General URI Policies](#)
- [Course Communications](#)

Notes

- [1. Welcome and Introduction](#)

Assignments

- [1. Assignments](#)

Portfolio

- [Portfolio](#)
- [Formatting Tips](#)

FAQ

- [FAQ](#)
- [Syllabus and Grading FAQ](#)
- [Git and GitHub](#)
- [Code Errors](#)

Resources

- [Glossary](#)
- [References on Python](#)
- [Cheatsheet](#)
- [Data Sources](#)
- [General Tips and Resources](#)
- [How to Study in this class](#)
- [Getting Help with Programming](#)
- [Terminals and Environments](#)
- [Getting Organized for class](#)
- [Advice from FA2020 Students](#)
- [Advice from FA2021 Students](#)
- [Letters to Future students](#)

Welcome to the course manual for CSC310 at URI with Professor Brown.

This class meets TTh 5-6:15pm in Tyler 108.

This website will contain the syllabus, class notes, and other reference material for the class.

[Course Calendar on BrightSpace](#)

Tip

[subscribe to that calendar](#) in your favorite calendar application

Navigating the Sections

The Syllabus section has logistical operations for the course broken down into sections. You can also read straight through by starting in the first one and navigating to the next section using the arrow navigation at the end of the page.

This site is a resource for the course. We do not follow a text book for this course, but all notes from class are posted in the notes section, accessible on the left hand side menu, visible on large screens and in the menu on mobile.

The resources section has links and short posts that provide more context and explanation. Content in this section is for the most part not strictly the material that you'll be graded on, but it is often material that will help you understand and grow as a programmer and data scientist.

Reading each page

All class notes can be downloaded in multiple formats, including as a notebook. Some pages of the syllabus and resources are also notebooks, if you want to see behind the curtain of how I manage the course information.

Try it Yourself

Notes will have exercises marked like this

Question from Class

Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Long answers will be in the main notes

Further reading

Notes that are mostly links to background and context will be highlighted like this. These are optional, but will mostly help you understand code excerpts they relate to.

Hint

Both notes and assignment pages will have hints from time to time. Pay attention to these on the notes, they'll typically relate to things that will appear in the assignment.

Think Ahead

Think ahead boxes will guide you to start thinking about what can go into your portfolio to build on the material at hand.

Click here!

Special tips will be formatted like this

Question from class

Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Short questions will be in the margin note



Check your Comprehension

Questions to use to check your comprehension will look like this

About

About the topic

Data science exists at the intersection of computer science, statistics, and domain expertise. That means writing programs to access and manipulate data so that it becomes available for analysis using statistical and machine learning techniques is at the core of data science. Data scientists use their data and analytical ability to find and interpret rich data sources; manage large amounts of data despite hardware, software, and bandwidth constraints; merge data sources; ensure consistency of datasets; create visualizations to aid in understanding data; build mathematical models using the data; and present and communicate the data insights/findings.

About the goals and preparation

This course provides a survey of data science. Topics include data driven programming in Python; data sets, file formats and meta-data; descriptive statistics, data visualization, and foundations of predictive data modeling and machine learning; accessing web data and databases; distributed data management. You will work on weekly programming problems such as accessing data in database and visualize it or build machine learning models of a given data set.

Basic programming skills (CSC201 or CSC211) are a prerequisite to this course. This course is a prerequisite course to machine learning, where you learn how machine learning algorithms work. In this course, we will start with a very fast review of basic programming ideas, since you've already done that before. We will learn how to *use* machine learning algorithms to do data science, but not how to *build* machine learning algorithms, we'll use packages that implement the algorithms for us.

About the course

This course is designed to make you a better programmer while learning data science. You may be stronger in one of those areas than the other at the beginning, but you should grow in both areas either way by the end of the semester.

About this syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly online at rhodyprog4ds.github.io/BrownFall20/syllabus. If you choose to download a copy of it, note that it is only a copy. You can get notification of changes from GitHub by "watching" the [repository](#). You can view the date of changes and exactly what changes were made on the Github [commits](#) page.

Creating an [issue on the repository](#) is also a good way to ask questions about anything in the course it will prompt additions and expand the FAQ section.

About your instructor

Name: Dr. Sarah Brown Office hours: TBA via zoom, link in BrightSpace

Dr. Brown is an Assistant Professor of Computer Science, who does research on how social context changes machine learning. Dr. Brown earned a PhD in Electrical Engineering from Northeastern University, completed a postdoctoral fellowship at University of California Berkeley, and worked as a postdoctoral research associate at Brown University before joining URI. At Brown University, Dr. Brown taught the Data and Society course for the Master's in Data Science Program.

Important

For assignment or notes specific issues, a comment on the corresponding repository is the best. I cannot help you with code issues from screenshots.

Note

Whether you use CSC or DSP does not matter.

The best way to contact me for general questions is e-mail or by dropping into my office hours. Please include [\[CSC310\]](#) or [\[DSP310\]](#) in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it. I rarely check e-mail between 6pm and 9am, on weekends or holidays. You might see me post or send things during these hours, but I will not reliably see emails that arrive during those hours.

Tools and Resources

We will use a variety of tools to conduct class and to facilitate your programming. You will need a computer with Linux, MacOS, or Windows. It is unlikely that a tablet will be able to do all of the things required in this course. A Chromebook may work, especially with developer tools turned on. Ask Dr. Brown if you need help getting access to an adequate computer.

All of the tools and resources below are either:

- paid for by URI **OR**
- freely available online.

BrightSpace

This will be the central location from which you can access all other materials. Any links that are for private discussion among those enrolled in the course will be available only from our course [Brightspace site](#).

Prismia chat

Our class link for [Prismia chat](#) is available on Brightspace. We will use this for chatting and in-class understanding checks.

On Prismia, all students see the instructor's messages, but only the Instructor and TA see student responses.

Course website

The course manual will have content including the class policies, scheduling, class notes, assignment information, and additional resources. This will be linked from Brightspace and available publicly online at rhodyprog4ds.github.io/BrownSpring23/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub

You will need a [GitHub](#) Account. If you do not already have one, please [create one](#) by the first day of class. If you have one, but have not used it recently, you may need to update your password and login credentials as the [Authentication rules](#) changed over the summer. In order to use the command line with https, you will need to [create a Personal Access Token](#) for each device you use. In order to use the command line with SSH, set up your public key.

Programming Environment

Important

TL;DR [\[1\]](#)

- check Brightspace
- Log in to Prismia Chat
- Make a GitHub Account
- Install Python
- Install Git

Note

Seeing the BrightSpace site requires logging in with your URI SSO and being enrolled in the course

This is a programming course, so you will need a programming environment. In order to complete assignments you need the items listed in the requirements list. The easiest way to meet these requirements is to follow the recommendations below. I will provide instruction assuming that you have followed the recommendations.

Requirements:

- Python with scientific computing packages (numpy, scipy, jupyter, pandas, seaborn, sklearn)
- [Git](#)
- A web browser compatible with [Jupyter Notebooks](#)

⚠ Warning

Everything in this class will be tested with the up to date (or otherwise specified) version of Jupyter Notebooks. Google Colab is similar, but not the same, and some things may not work there. It is an okay backup, but should not be your primary work environment.

📌 Note

all Git instructions will be given as instructions for the command line interface and GitHub specific instructions via the web interface. You may choose to use GitHub desktop or built in IDE tools, but the instructional team may not be able to help.

Recommendation:

- Install python via [Anaconda](#)
- if you use Windows, install Git with [GitBash](#) ([video instructions](#)).
- if you use MacOS, install Git with the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this by trying to run git from the Terminal the very first time.`git --version`
- if you use Chrome OS, follow these instructions:

1. Find Linux (Beta) in your settings and turn that on.
2. Once the download finishes a Linux terminal will open, then enter the commands: `sudo apt-get update` and `sudo apt-get upgrade`. These commands will ensure you are up to date.
3. Install tmux with:

```
sudo apt -t stretch-backports install tmux
```

4. Next you will install nodejs, to do this, use the following commands:

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash
sudo apt-get install -y nodejs
sudo apt-get install -y build-essential.
```

5. Next install Anaconda's Python from the website provided by the instructor and use the top download link under the Linux options.
6. You will then see a .sh file in your downloads, move this into your Linux files.
7. Make sure you are in your home directory (something like home/YOURUSERNAME), do this by using the `pwd` command.
8. Use the `bash` command followed by the file name of the installer you just downloaded to start the installation.
9. Next you will add Anaconda to your Linux PATH, do this by using the `vim .bashrc` command to enter the .bashrc file, then add the `export PATH=/home/YOURUSERNAME/anaconda3/bin/:$PATH` line. This can be placed at the end of the file.
10. Once that is inserted you may close and save the file, to do this hold escape and type `:x`, then press enter. After doing that you will be returned to the terminal where you will then type the `source .bashrc` command.
11. Next, use the `jupyter notebook --generate-config` command to generate a Jupyter Notebook.
12. Then just type `jupyter lab` and a Jupyter Notebook should open up.

Optional:

- Text Editor: you may want a text editor outside of the Jupyter environment. Jupyter can edit markdown files (that you'll need for your portfolio), in browser, but it is more common to use a text editor like Atom or Sublime for this purpose.

Video install instructions for Anaconda:

- [Windows](#)
- [Mac](#)

On Mac, to install python via environment, [this article may be helpful](#)

- I don't have a video for linux, but it's a little more straight forward.

Textbook

The text for this class is a reference book and will not be a source of assignments. It will be a helpful reference and you may be directed there for answers to questions or alternate explanations of topics.

Python for Data Science is available free [online](#):

Zoom (backup and office hours only)

This is where we will meet if for any reason we cannot be in person. You will find the link to class zoom sessions on Brightspace.

URI provides all faculty, staff, and students with a paid Zoom account. It *can* run in your browser or on a mobile device, but you will be able to participate in class best if you download the [Zoom client](#) on your computer. Please [log in](#) and [configure your account](#). Please add a photo of yourself to your account so that we can still see your likeness in some form when your camera is off. You may also wish to use a virtual background and you are welcome to do so.

Class will be interactive, so if you cannot be in a quiet place at class time, headphones with a built in microphone are strongly recommended.

For help, you can access the [instructions provided by IT](#).

[1] Too long; didn't read.

Data Science Achievements

In this course there are 5 learning outcomes that I expect you to achieve by the end of the semester. To get there, you'll focus on 15 smaller achievements that will be the basis of your grade. This section will describe how the topics covered, the learning outcomes, and the achievements are covered over time. In the next section, you'll see how these achievements turn into grades.

Learning Outcomes

By the end of the semester

1. (process) Describe the process of data science, define each phase, and identify standard tools
2. (data) Access and combine data in multiple formats for analysis
3. (exploratory) Perform exploratory data analyses including descriptive statistics and visualization
4. (modeling) Select models for data by applying and evaluating multiple models to a single dataset
5. (communicate) Communicate solutions to problems with data in common industry formats

We will build your skill in the **process** and **communicate** outcomes over the whole semester. The middle three skills will correspond roughly to the content taught for each of the first three portfolio checks.

Schedule

The course will meet TTh 5-6:15pm in Tyler 108. Every class will include participatory live coding (instructor types code while explaining, students follow along)) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

Programming assignments that will be due each week Tuesday by 11:59pm.

week	topics	skills
1	[admin, python review]	process
2	Loading data, Python review	[access, prepare, summarize]
3	Exploratory Data Analysis	[summarize, visualize]
4	Data Cleaning	[prepare, summarize, visualize]
5	Databases, Merging DataFrames	[access, construct, summarize]
6	Modeling, classification performance metrics, cross validation	[evaluate]
7	Naive Bayes, decision trees	[classification, evaluate]
8	Regression	[regression, evaluate]
9	Clustering	[clustering, evaluate]
10	SVM, parameter tuning	[optimize, tools]
11	KNN, Model comparison	[compare, tools]
12	Text Analysis	[unstructured]
13	Images Analysis	[unstructured, tools]
14	Deep Learning	[tools, compare]

Note

On the [Course Calendar on BrightSpace](#) page you can get a feed link to add to the calendar of your choice by clicking on the subscribe (star) button on the top right of the page. Class is for 1 hour there because of Brightspace/zoom integration limitations, but that calendar includes the zoom link.

Achievement Definitions

The table below describes how your participation, assignments, and portfolios will be assessed to earn each achievement. The keyword for each skill is a short name that will be used to refer to skills throughout the course materials; the full description of the skill is in this table.

keyword		skill	Level 1	Level 2	Level 3
	python	pythonic code writing	python code that mostly runs, occasional pep8 adherence	python code that reliably runs, frequent pep8 adherence	reliable, efficient, pythonic code that consistently adheres to pep8
	process	describe data science as a process	Identify basic components of data science	Describe and define each stage of the data science process	Compare different ways that data science can facilitate decision making
	access	access data in multiple formats	load data from at least one format; identify the most common data formats	Load data for processing from the most common formats; Compare and contrast most common formats	access data from both common and uncommon formats and identify best practices for formats in different contexts
	construct	construct datasets from multiple sources	identify what should happen to merge datasets or when they can be merged	apply basic merges	merge data that is not automatically aligned
	summarize	Summarize and describe data	Describe the shape and structure of a dataset in basic terms	compute summary statistics of a whole dataset and grouped data	Compute and interpret various summary statistics of subsets of data
	visualize	Visualize data	identify plot types, generate basic plots from pandas	generate multiple plot types with complete labeling with pandas and seaborn	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
	prepare	prepare data for analysis	identify if data is or is not ready for analysis, potential problems with data	apply data reshaping, cleaning, and filtering as directed	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
	evaluate	Evaluate model performance	Explain basic performance metrics for different data science tasks	Apply and interpret basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
	classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit, apply, and interpret preselected classification model to a dataset	fit and apply classification models and select appropriate classification models for different contexts
	regression	Apply Regression	identify what data that can be used for regression looks like	fit and interpret linear regression models	fit and explain regularized or nonlinear regression
	clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results

keyword	skill	Level 1	Level 2	Level 3
optimize	Optimize model parameters	Identify when model parameters need to be optimized	Optimize basic model parameters such as model order	Select optimal parameters based of mutiple quantiative criteria and automate parameter tuning
compare	compare models	Qualitatively compare model classes	Compare model classes in specific terms and fit models in terms of traditional model performance metrics	Evaluate tradeoffs between different model comparison types
representation	Choose representations and transform data	Identify options for representing text and categorical data in many contexts	Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred fully specified problems with a single tool pipeline	Solve well-structred, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

Assignments and Skills

Using the keywords from the table above, this table shows which assignments you will be able to demonstrate which skills and the total number of assignments that assess each skill. This is the number of opportunities you have to earn Level 2 and still preserve 2 chances to earn Level 3 for each skill.

keyword	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
python	1	1	0	1	1	0	0	0	0	0	0	0	0	4
process	1	0	0	0	0	1	1	1	1	1	1	0	0	7
access	0	1	1	1	1	0	0	0	0	0	0	0	0	4
construct	0	0	0	0	1	0	1	1	0	0	0	0	0	3
summarize	0	0	1	1	1	1	1	1	1	1	1	1	1	11
visualize	0	0	1	1	0	1	1	1	1	1	1	1	1	10
prepare	0	0	0	1	1	0	0	0	0	0	0	0	0	2
evaluate	0	0	0	0	0	1	1	1	0	1	1	0	0	5
classification	0	0	0	0	0	0	1	0	0	1	0	0	0	2
regression	0	0	0	0	0	0	0	1	0	0	1	0	0	2
clustering	0	0	0	0	0	0	0	0	1	0	1	0	0	2
optimize	0	0	0	0	0	0	0	0	0	1	1	0	0	2
compare	0	0	0	0	0	0	0	0	0	0	1	0	1	2
representation	0	0	0	0	0	0	0	0	0	0	0	1	1	2
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	4

Warning

process achievements are accumulated a little slower. Prior to portfolio check 1, only level 1 can be earned. Portfolio check 1 is the first chance to earn level 2 for process, then level 3 can be earned on portfolio check 2 or later.

Portfolios and Skills

The objective of your portfolio submissions is to earn Level 3 achievements. The following table shows what Level 3 looks like for each skill and identifies which portfolio submissions you can earn that Level 3 in that skill.

		Level 3	P1	P2	P3	P4
keyword						
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	1	
process	Compare different ways that data science can facilitate decision making	0	1	1	1	
access	access data from both common and uncommon formats and identify best practices for formats in different contexts	1	1	0	1	
construct	merge data that is not automatically aligned	1	1	0	1	
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	1	
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters	1	1	0	1	
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	1	
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	1	
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	1	
regression	fit and explain regularized or nonlinear regression	0	1	1	1	
clustering	apply multiple clustering techniques, and interpret results	0	1	1	1	
optimize	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning	0	0	1	1	
compare	Evaluate tradeoffs between different model comparison types	0	0	1	1	
representation	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance	0	0	1	1	
workflow	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools	0	0	1	1	

Grading

This section of the syllabus describes the principles and mechanics of the grading for the course. This course will be graded on a basis of a set of *skills* (described in detail the next section of the syllabus). This is in contrast to more common grading on a basis of points earned through assignments.

Principles of Grading

Learning happens through practice and feedback. My goal as a teacher is for you to learn. The grading in this course is based on your learning of the material, rather than your completion of the activities that are assigned.

This course is designed to encourage you to work steadily at learning the material and demonstrating your new knowledge. There are no single points of failure, where you lose points that cannot be recovered. Also, you cannot cram anything one time and then forget it. The material will build and you have to demonstrate that you retained things.

- Earning a C in this class means you have a general understanding of Data Science and could participate in a basic conversation about all of the topics we cover. I expect everyone to reach this level.
- Earning a B means that you could solve simple data science problems on your own and complete parts of more complex problems as instructed by, for example, a supervisor in an internship or entry level job. This is a very accessible goal, it does not require you to get anything on the first try or to explore topics on your own. I expect most students to reach this level.
- Earning an A means that you could solve moderately complex problems independently and discuss the quality of others' data science solutions. This class will be challenging, it requires you to explore topics a little deeper than we cover them in class, but unlike typical grading it does not require all of your assignments to be near perfect.

Grading this way also is more amenable to the fact that there are correct and incorrect ways to do things, but there is not always a single correct answer to a realistic data science problem. Your work will be assessed on whether or not it demonstrates your learning of the targeted skills. You will also receive feedback on how to improve.

How it works

There are 15 skills that you will be graded on in this course. While learning these skills, you will work through a progression of learning. Your grade will be based on earning 45 achievements that are organized into 15 skill groups with 3 levels for each.

These map onto letter grades roughly as follows:

- If you achieve level 1 in all of the skills, you will earn at least a C in the course.
- To earn a B, you must earn all of the level 1 and level 2 achievements.
- To earn an A, you must earn all of the achievements.

You will have at least three opportunities to earn every level 2 achievement. You will have at least two opportunities to earn every level 3 achievement. You will have three *types* of opportunities to demonstrate your current skill level: participation, assignments, and a portfolio.

Each level of achievement corresponds to a phase in your learning of the skill:

- To earn level 1 achievements, you will need to demonstrate basic awareness of the required concepts and know approximately what to do, but you may need specific instructions of which things to do or to look up examples to modify every step of the way. You can earn level 1 achievements in class, assignments, or portfolio submissions.
- To earn level 2 achievements you will need to demonstrate understanding of the concepts and the ability to apply them with instruction after earning the level 1 achievement for that skill. You can earn level 2 achievements in assignments or portfolio submissions.
- To earn level 3 achievements you will be required to consistently execute each skill and demonstrate deep understanding of the course material, after achieving level 2 in that skill. You can earn level 3 achievements only through your portfolio submissions.

For each skill these are defined in the [Achievement Definition Table](#)

Participation

While attending synchronous class sessions, there will be understanding checks and in class exercises. Completing in class exercises and correctly answering questions in class can earn level 1 achievements. In class questions will be administered through the classroom chat platform Prismia.chat; these records will be used to update your skill progression. You can also earn level 1 achievements from adding annotation to a section of the class notes.

Assignments

For your learning to progress and earn level 2 achievements, you must practice with the skills outside of class time.

Assignments will each evaluate certain skills. After your assignment is reviewed, you will get qualitative feedback on your work, and an assessment of your demonstration of the targeted skills.

Portfolio Checks

To earn level 3 achievements, you will build a portfolio consisting of reflections, challenge problems, and longer analyses over the course of the semester. You will submit your portfolio for review 4 times. The first two will cover the skills taught up until 1 week before the submission deadline.

The third and fourth portfolio checks will cover all of the skills. The fourth will be due during finals. This means that, if you have achieved mastery of all of the skills by the 3rd portfolio check, you do not need to submit the fourth one.

Portfolio prompts will be given throughout the class, some will be structured questions, others may be questions that arise in class, for which there is not time to answer.

TLDR

You *could* earn a C through in class participation alone, if you make nearly zero mistakes. To earn a B, you must complete assignments and participate in class. To earn an A you must participate, complete assignments, and build a portfolio.

Detailed mechanics

On Brightspace there are 45 Grade items that you will get a 0 or a 1 grade for. These will be revealed, so that you can view them as you have an opportunity to demonstrate each one. The table below shows the minimum number of skills at each level to earn each letter grade.

	Level 3	Level 2	Level 1
letter grade			
A	15	15	15
A-	10	15	15
B+	5	15	15
B	0	15	15
B-	0	10	15
C+	0	5	15
C	0	0	15
C-	0	0	10
D+	0	0	5
D	0	0	3

For example, if you achieve level 2 on all of the skills and level 3 on 7 skills, that will be a B+.

If you achieve level 3 on 14 of the skills, but only level 1 on one of the skills, that will be a B-, because the minimum number of level 2 achievements for a B is 15. In this scenario the total number of achievements is 14 at level 3, 14 at level 2 and 15 at level 3, because you have to earn achievements within a skill in sequence.

The letter grade can be computed as follows

Important

this will be revealed after assignment 1

Warning

If you will skip an assignment, please accept the GitHub assignment and then close the Feedback pull request with a comment. This way we can make sure that you have support you need.

Note

In this example, you will have also achieved level 1 on all of the skills, because it is a prerequisite to level 2.

For example you can run the code like this in a cell to see the output

```
compute_grade(15, 15, 15)
```

```
'A'
```

```
compute_grade(14, 14, 14)
```

```
'C-'
```

Or use `assert` to test it formally

```
assert compute_grade(14, 14, 14) == 'C-'
```

```
assert compute_grade(15, 15, 15) == 'A'
```

```
assert compute_grade(15, 15, 11) == 'A-'
```

Late work

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt your grade. If you do not submit any assignments that cover a given skill, you may earn the level 2 achievement in that skill through a portfolio check, but you will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most assignments cover multiple skills, so partially completing the assignment may earn level 2 for one, but not all. Submitting *something* even if it is not perfect is important to keeping conversation open and getting feedback and help continuously.

Building your Data Science Portfolio should be an ongoing process, where you commit work to your portfolio frequently. If something comes up and you cannot finish all that you would like assessed by the deadline, open an [Extension Request](#) issue on your repository.

In this issue, include:

1. A new deadline proposal
2. What additional work you plan to add
3. Why the extension is important to your learning
4. Why the extension will not hinder your ability to complete the next assignment on time.

This request should be no more than 7 sentences.

Portfolio due dates will be announced well in advance and prompts for it will be released weekly. You should spend some time working on it each week, applying what you've learned so far, from the feedback on previous assignments.

Grading Examples

If you always attend and get everything correct, you will earn an A and you won't need to submit the 4th portfolio check.

Getting an A Without Perfection

Note

You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Map to an A

How Achievements were earned

	Level 1	Level 2	Level 3
python	A1	A3	P1
process	A1	P1	P2
access	2	A2	P1
construct	5	A5	P1
summarize	3	A3	P1
visualize	3	A3	P2
prepare	4	A5	P2
classification	A10	P2	P3
regression	8	A11	P2
clustering	9	A9	P3
evaluate	7	A11	P3
optimize	10	A11	P4
compare	11	A13	P3
unstructured	12	A13	P4
tools	11	A13	P3

Activity Legend

In class

Assignment

Portfolio Check



Other Activities



Attended, but did not understand



Submitted, but incorrect



Missed class



Not submitted



Submitted, but incorrect



Not submitted



Not submitted



Attended, but all level 1 complete



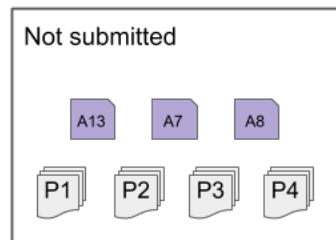
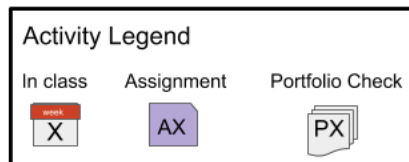
Attended, but all level 1 complete

In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the **python**, **process**, and **classification** skills, the level 1 achievements were earned on assignments, not in class. For the **process** and **classification** skills, the level 2 achievements were not earned on assignments, only on portfolio checks, but they were earned on the first portfolio of those skills, so the level 3 achievements were earned on the second portfolio check for that skill. This student's fourth portfolio only demonstrated two skills: **optimize** and **unstructured**. It included only 1 analysis, a text analysis with optimizing the parameters of the model. Assignments 4 and 7 were both submitted, but didn't earn any achievements, the student got feedback though, that they were able to apply in later assignments to earn the achievements. The student missed class week 6 and chose to not submit assignment 6 and use week 7 to catch up. The student had too much work in another class and chose to skip assignment 8. The student tried assignment 12, but didn't finish it on time, so it was not graded, but the student visited office hours to understand and be sure to earn the level 2 **unstructured** achievement on assignment 13.

Getting a B with minimal work

Map to a B easily

	Level 1	Level 2	Level 3
python			
process			
access			
construct			
summarize			
visualize			
prepare			
classification			
regression			
clustering			
evaluate			
optimize			
compare			
unstructured			
tools			

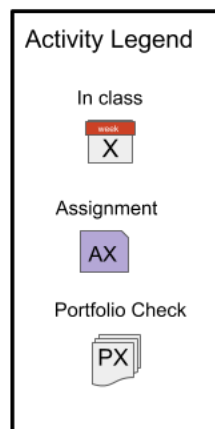


In this example, the student earned all level 1 achievements in class and all level 2 on assignments. This student was content with getting a B and chose to not submit a portfolio.

Getting a B while having trouble

Map to a B, having trouble

	Level 1	Level 2	Level 3
python			
process			
access			
construct			
summarize			
visualize			
prepare			
classification			
regression			
clustering			
evaluate			
optimize			
compare			
unstructured			
tools			



In this example, the student struggled to understand in class and on assignments. Assignments were submitted that showed some understanding, but all had some serious mistakes, so only level 1 achievements were earned from assignments. The student wanted to get a B and worked hard to get the level 2 achievements on the portfolio checks.

Grading Policies

Late Work

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt your grade. If you do not submit any assignments that cover a given skill, you may earn the level 2 achievement in that skill through a portfolio check, but you will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most assignments cover multiple skills, so partially completing the assignment may earn level 2 for one, but not all. Submitting *something* even if it is not perfect is important to keeping conversation open and getting feedback and help continuously.

Building your Data Science Portfolio should be an ongoing process, where you commit work to your portfolio frequently. If something comes up and you cannot finish all that you would like assessed by the deadline, open an [Extension Request](#) issue on your repository.

In this issue, include:

1. A new deadline proposal
2. What additional work you plan to add
3. Why the extension is important to your learning
4. Why the extension will not hinder your ability to complete the next assignment on time.

This request should be no more than 7 sentences.

Portfolio due dates will be announced well in advance and prompts for it will be released weekly. You should spend some time working on it each week, applying what you've learned so far, from the feedback on previous assignments.

Regrading

Re-request a review on your Feedback Pull request.

For general questions, post on the conversation tab of your Feedback PR with your request.

For specific questions, reply to a specific comment.

If you think we missed *where* you did something, add a comment on that line (on the code tab of the PR, click the plus (+) next to the line) and then post on the conversation tab with an overview of what you're requesting and tag @brownsarahm

Support

Warning

URI changed some links and this page is not yet up to date

Note

You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Academic Enhancement Center

Academic Enhancement Center (for undergraduate courses): Located in Roosevelt Hall, the AEC offers free face-to-face and web-based services to undergraduate students seeking academic support. Peer tutoring is available for STEM-related courses by appointment online and in-person. The Writing Center offers peer tutoring focused on supporting undergraduate writers at any stage of a writing assignment. The UCS160 course and academic skills consultations offer students strategies and activities aimed at improving their studying and test-taking skills. Complete details about each of these programs, up-to-date schedules, contact information and self-service study resources are all available on the [AEC website](#).

- **STEM Tutoring** helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses. The STEM Tutoring program offers free online and limited in-person peer-tutoring this fall. Undergraduates in introductory STEM courses have a variety of small

group times to choose from and can select occasional or weekly appointments. Appointments and locations will be visible in the TutorTrac system on September 14th, 2020. The TutorTrac application is available through [URI Microsoft 365 single sign-on](#) and by visiting [aec.uri.edu](#). More detailed information and instructions can be found on the [AEC tutoring page](#).

- **Academic Skills Development** resources helps students plan work, manage time, and study more effectively. In Fall 2020, all Academic Skills and Strategies programming are offered both online and in-person. UCS160: Success in Higher Education is a one-credit course on developing a more effective approach to studying. Academic Consultations are 30-minute, 1 to 1 appointments that students can schedule on Starfish with Dr. David Hayes to address individual academic issues. Study Your Way to Success is a self-guided web portal connecting students to tips and strategies on studying and time management related topics. For more information on these programs, visit the [Academic Skills Page](#) or contact Dr. Hayes directly at davidhayes@uri.edu.
- The **Undergraduate Writing Center** provides free writing support to students in any class, at any stage of the writing process: from understanding an assignment and brainstorming ideas, to developing, organizing, and revising a draft. Fall 2020 services are offered through two online options: 1) real-time synchronous appointments with a peer consultant (25- and 50-minute slots, available Sunday - Friday), and 2) written asynchronous consultations with a 24-hour turn-around response time (available Monday - Friday). Synchronous appointments are video-based, with audio, chat, document-sharing, and live captioning capabilities, to meet a range of accessibility needs. View the synchronous and asynchronous schedules and book online, visit uri.mywconline.com.

General URI Policies

Warning

URI changed some links and this page is not yet up to date

Anti-Bias Statement:

We respect the rights and dignity of each individual and group. We reject prejudice and intolerance, and we work to understand differences. We believe that equity and inclusion are critical components for campus community members to thrive. If you are a target or a witness of a bias incident, you are encouraged to submit a report to the URI Bias Response Team at www.uri.edu/brt. There you will also find people and resources to help.

Disability Services for Students Statement:

Your access in this course is important. Please send me your Disability Services for Students (DSS) accommodation letter early in the semester so that we have adequate time to discuss and arrange your approved academic accommodations. If you have not yet established services through DSS, please contact them to engage in a confidential conversation about the process for requesting reasonable accommodations in the classroom. DSS can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dss@etal.uri.edu. We are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

Students are expected to be honest in all academic work. A student's name on any written work, quiz or exam shall be regarded as assurance that the work is the result of the student's own independent thought and study. Work should be stated in the student's own words, properly attributed to its source. Students have an obligation to know how to quote, paraphrase, summarize, cite and reference the work of others with integrity. The following are examples of academic dishonesty.

- Using material, directly or paraphrasing, from published sources (print or electronic) without appropriate citation

- Claiming disproportionate credit for work not done independently
- Unauthorized possession or access to exams
- Unauthorized communication during exams
- Unauthorized use of another's work or preparing work for another student
- Taking an exam for another student
- Altering or attempting to alter grades
- The use of notes or electronic devices to gain an unauthorized advantage during exams
- Fabricating or falsifying facts, data or references
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

URI COVID-19 Statement

The University is committed to delivering its educational mission while protecting the health and safety of our community. While the university has worked to create a healthy learning environment for all, it is up to all of us to ensure our campus stays that way.

As members of the URI community, students are required to comply with standards of conduct and take precautions to keep themselves and others safe. Visit web.uri.edu/coronavirus/ for the latest information about the URI COVID-19 response.

- [Universal indoor masking](#) is required by all community members, on all campuses, regardless of vaccination status. If the universal mask mandate is discontinued during the semester, students who have an approved exemption and are not fully vaccinated will need to continue to wear a mask indoors and maintain physical distance.
- Students who are experiencing symptoms of illness should not come to class. Please stay in your home/room and notify URI Health Services via phone at 401-874-2246.
- If you are already on campus and start to feel ill, go home/back to your room and self-isolate. Notify URI Health Services via phone immediately at 401-874-2246.

If you are unable to attend class, please notify me at brownsarahm@uri.edu. We will work together to ensure that course instruction and work is completed for the semester.

Course Communications

Announcements

Announcements will be made via GitHub Release. You can view them [online in the releases page](#) or you can get notifications by watching the repository, choosing "Releases" under custom [see GitHub docs for instructions with screenshots](#). You can choose GitHub only or e-mail notification [from the notification settings page](#)

Help Hours

```
File "/tmp/ipykernel_1890/444382441.py", line 1
    help_df.style.hide(axis="index",
                                ^
SyntaxError: unexpected EOF while parsing
```

We have several different ways to communicate in this course. This section summarizes them

To reach out, By usage

usage	platform	area	note
in class	prismia	chat	outside of class time this is not monitored closely
any time	prismia	download transcript	use after class to get preliminary notes eg if you miss a class
private questions to your assignment	github	issue on assignment repo	eg bugs in your code"
for general questions that can help others	github	issue on course website	eg what the instructions of an assignment mean or questions about the syllabus
to share resources or ask general questions in a semi-private forum	github	discussion on community repo	include links in your portfolio
matters that don't fit into another category	e-mail	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Note

e-mail is last because it's not collaborative; other platforms allow us (Proessor + TA) to collaborate on who responds to things more easily.

By Platform

Use e-mail for

usage	area	note
matters that don't fit into another category	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Use github for

usage	area	note
private questions to your assignment	issue on assignment repo	eg bugs in your code"
for general questions that can help others	issue on course website	eg what the instructions of an assignment mean or questions about the syllabus
to share resources or ask general questions in a semi-private forum	discussion on community repo	include links in your portfolio

Use prismia for


usage	area	note
in class	chat	outside of class time this is not monitored closely
any time	download transcript	use after class to get preliminary notes eg if you miss a class

Tips

For assignment help

- **send in advance, leave time for a response** I check e-mail/github a small number of times per day, during work hours, almost exclusively. You might see me post to this site, post to BrightSpace, or comment on your assignments outside of my normal working hours, but I will not reliably see emails that arrive during those hours. This means that it is important to start assignments early.

Using issues

- use issues for content directly related to assignments. If you push your code to the repository and then open an issue, I can see your code and your question at the same time and download it to run it if I need to debug it
- use issues for questions about this syllabus or class notes. At the top right there's a GitHub logo  that allows you to open a issue (for a question) or suggest an edit (eg if you think there's a typo or you find an additional helpful resource related to something)

For E-mail

- use e-mail for general inquiries or notifications
- Please include `[CSC310]` or `[DSP310]` in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it.

Note

Whether you use CSC or DSP does not matter.

1. Welcome and Introduction

1.1. Prismia Chat

We will use these to monitor your participation in class and to gather information. Features:

- instructor only
- reply to you directly
- share responses for all

1.2. How this class will work

Participatory Live Coding

What is a topic you want to use data to learn about?

[Debugging is both technical and a soft skill](#)

1.3. Programming for Data Science vs other Programming

The audience is different, so the form is different.

In Data Science our product is more often a report than a program.

Note

Also, in data science we are *using code* to interact with data, instead of having a plan in advance

So programming for data science is more like *writing* it has a narrative flow and is made to be seen more than some other programming that you may have done.

Warning

Sometimes there will be points in the notes that were not made in class due to time or in response questions that came at the end of class.

1.4. Jupyter Notebooks

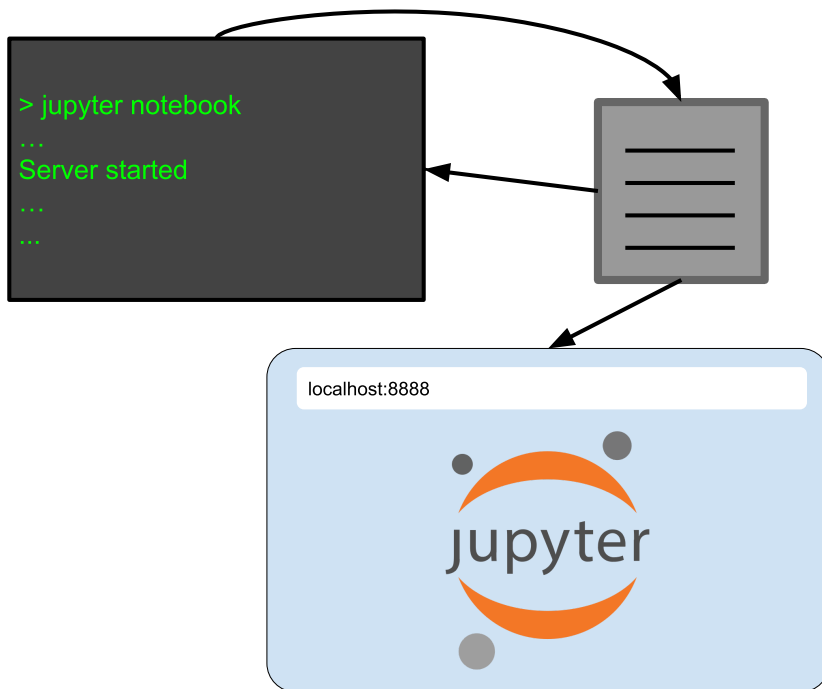
Launch a [jupyter notebook server](#):

- on Windows, use anaconda terminal
- on Mac/Linux, use terminal

```
cd path/to/where/you/save/notes
jupyter notebook
```

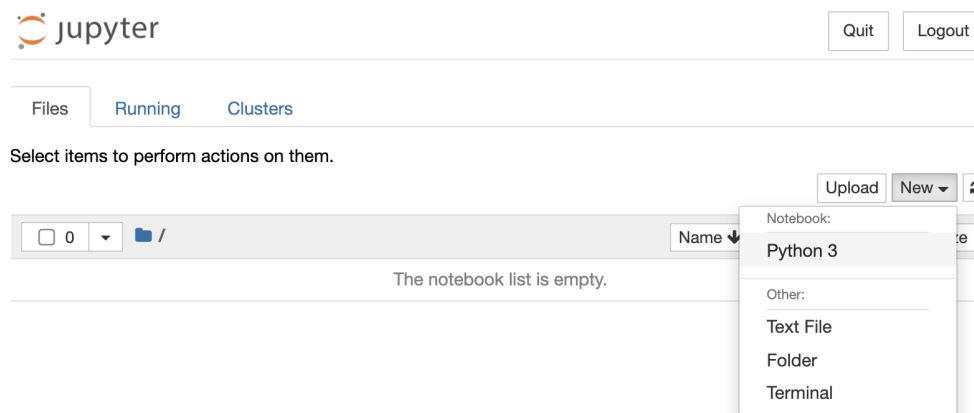
1.4.1. What just happened?

- launched a local web server
- opened a new browser tab pointed to it

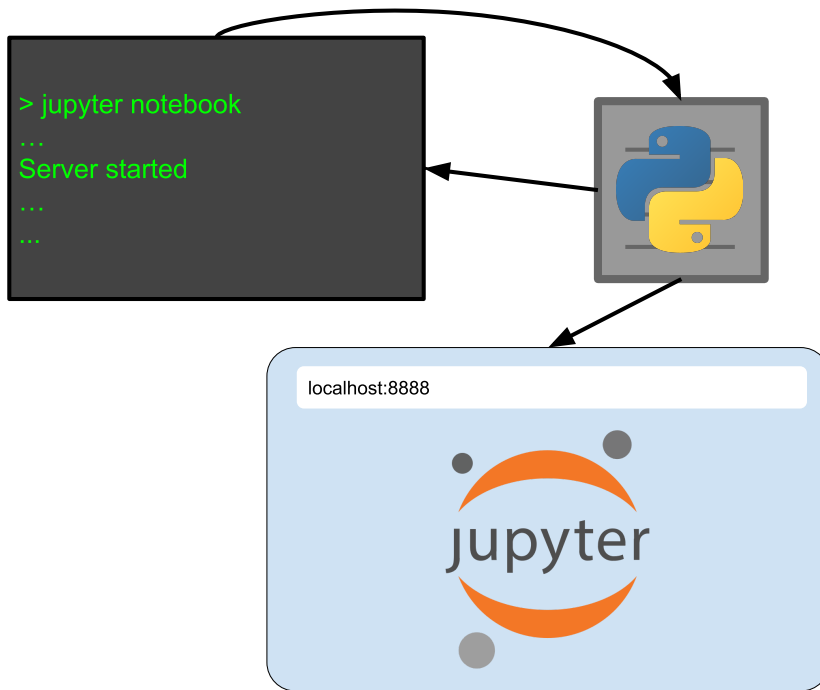


1.4.2. Start a Notebook

Go to the new menu in the top right and choose Python 3

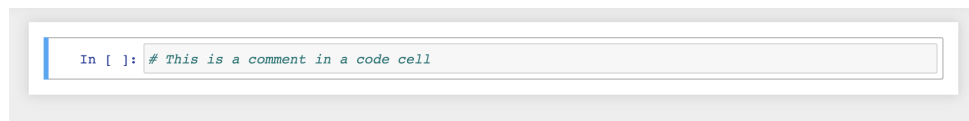


Now, it starts a python kernel on the webserver

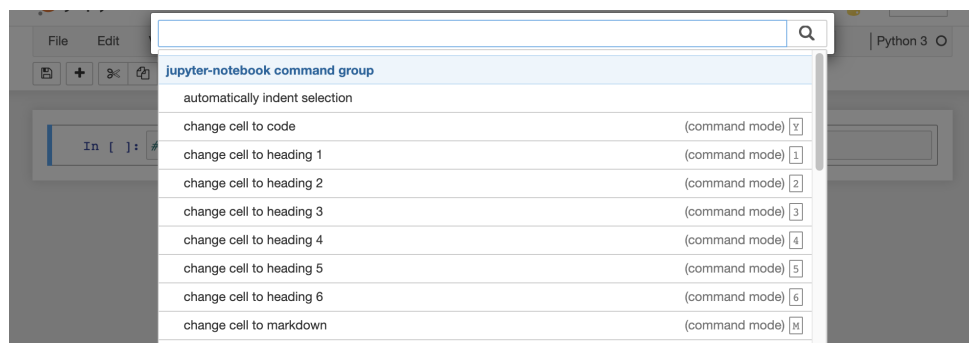


1.4.3. A jupyter notebook tour

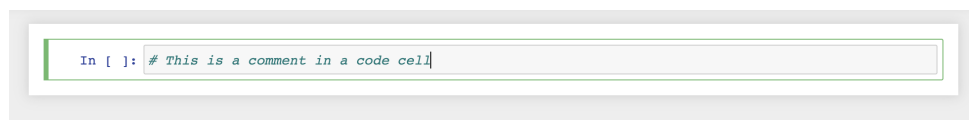
A Jupyter notebook has two modes. When you first open, it is in command mode. The border is blue in command mode.



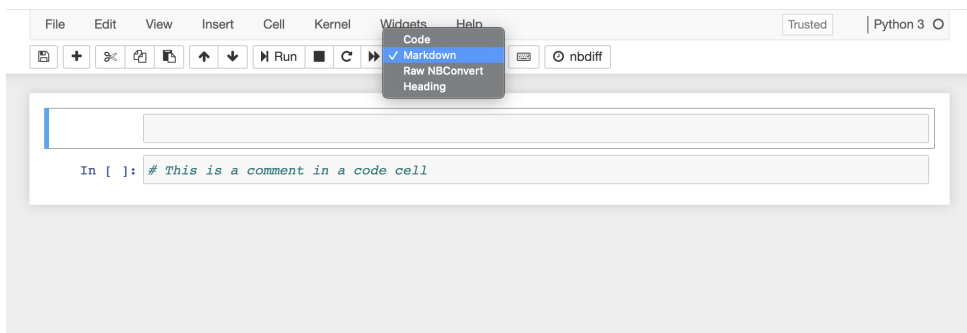
When you press a key in command mode it works like a shortcut. For example **p** shows the command search menu.



If you press **enter** (or **return**) or click on the highlighted cell, which is the boxes we can type in, it changes to edit mode. The border is green in edit mode



There are two type of cells that we will used: code and markdown. You can change that in command mode with **y** for code and **m** for markdown or on the cell type menu at the top of the notebook.



++

This is a markdown cell

- we can make
 - itemized lists of
 - bullet points
1. and we can make nubmered
 2. lists, and not have to worry
 3. about renumbering them
 4. if we add a step in the middle later

```
# this is a comment in a code cell , we can run this to see output
3+4
```

7

the output here is the value returned by the python interpreter for the last line of the cell

We set variables

```
name = 'sarah'
```

The notebook displays nothing when we do an assignment, bcause it returns nothing

name

'sarah'

we can put a variable there tosee it

```
course = 'csc310'
course
name
```

'sarah'

Note that this version doesn't show use the value for `course`

```
name = 'Sarah'
```

Important

In class, we ran these cells out of order and noticed how the value does not update unless we run the new version

name

```
'Sarah'
```

```
course
```

```
'csc310'
```

1.4.4. Notebook Reminders

Blue border is command mode, green border is edit mode

use Escape to get to command mode

Common command mode actions:

- m: switch cell to markdown
- y: switch cell to code
- a: add a cell above
- b: add a cell below
- c: copy cell
- v: paste the cell
- 0 + 0: restart kernel
- p: command menu

use enter/return to get to edit mode

In code cells, we can use a python interpreter, for example as a calculator.

```
4+6
```

```
10
```

It prints out the last line of code that it ran, even though it executes all of them

```
name = 'sarah'  
4+5  
name *3
```

```
'sarahsarahsarah'
```

1.5. Getting Help in Jupyter

Getting help is important in programming

When your cursor is inside the `()` of a function if you hold the shift key and press tab it will open a popup with information. If you press tab twice, it gets bigger and three times will make a popup window.

Python has a `print` function and we can use the help in jupyter to learn about how to use it in different ways.

```
print(name)
```

```
sarah
```

How do you use the `print` function to output: `Sarah_csc310`?

```
print(name, course, sep='_')
```

```
sarah_csc310
```



```
help(print)
```

Help on built-in function print in module builtins:

```
print(...)  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)  
  
    Prints the values to a stream, or to sys.stdout by default.  
    Optional keyword arguments:  
    file: a file-like object (stream); defaults to the current sys.stdout.  
    sep:  string inserted between values, default a space.  
    end:  string appended after the last value, default a newline.  
    flush: whether to forcibly flush the stream.
```

We can put as many values as we want there. That's what the `...` in the function signature means

```
print(name, course, 'hello', 'bye', sep='_')
```

```
sarah_csc310_hello_bye
```

```
print(name, course, 'hello', 'bye', sep='\n')
```

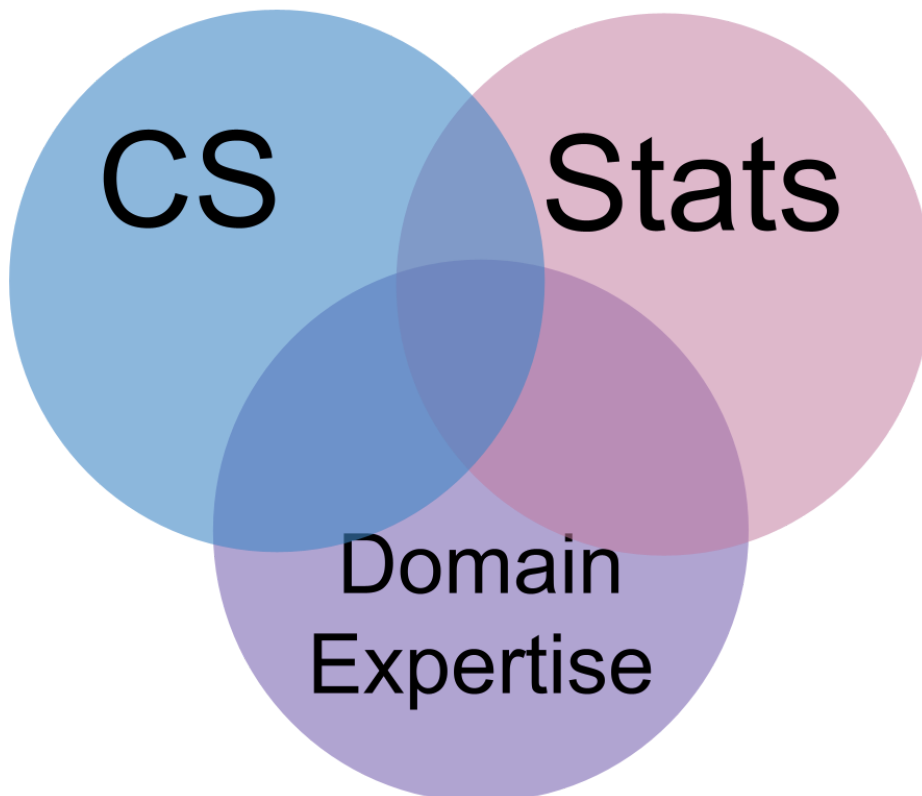
```
sarah  
csc310  
hello  
bye
```

Important

Basic programming is a prerequisite and we will go faster soon, but the goal of this review was to understand notebooks, getting help, and reading docstrings

1.6. What is Data Science?

Data Science is the combination of

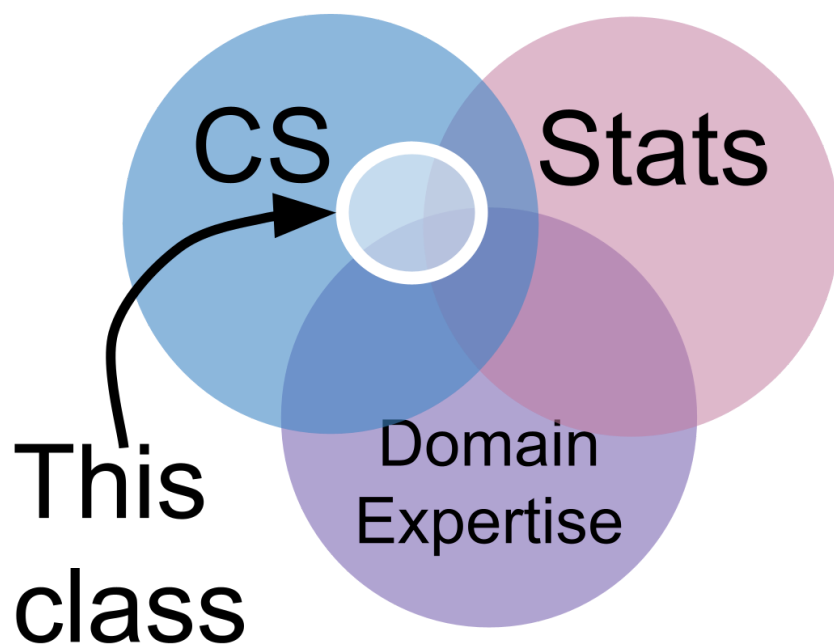


statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

computer science is so that we can manipulate visualize and automate the inferences we make.

domain expertise helps us have the intuition to know if what we did worked right. A statistic must be interpreted in context; the relevant context determines what they mean and which are valid. The context will say whether automating something is safe or not, it can help us tell whether our code actually worked right or not.

1.6.1. In this class,

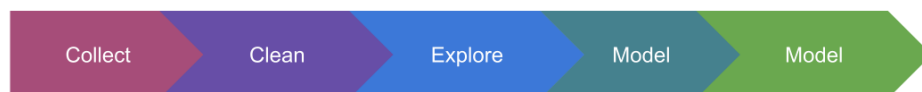


We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

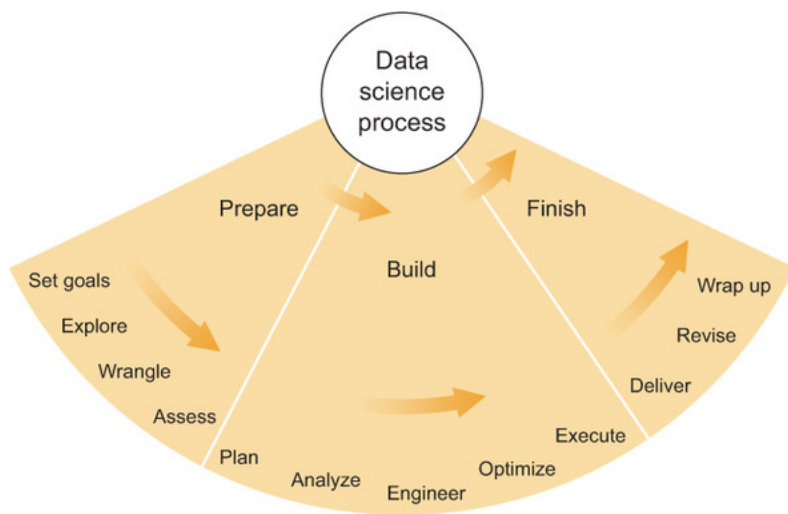
But there are many definitions. We'll use this one, but you may come across others.

1.6.2. How does data science happen?

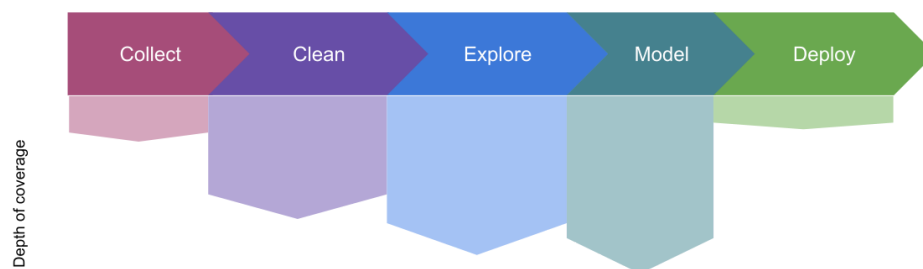
The most common way to think about what doing data science means is to think of this pipeline. It is in the perspective of the data, these are all of the things that happen to the data.



Another way to think about it

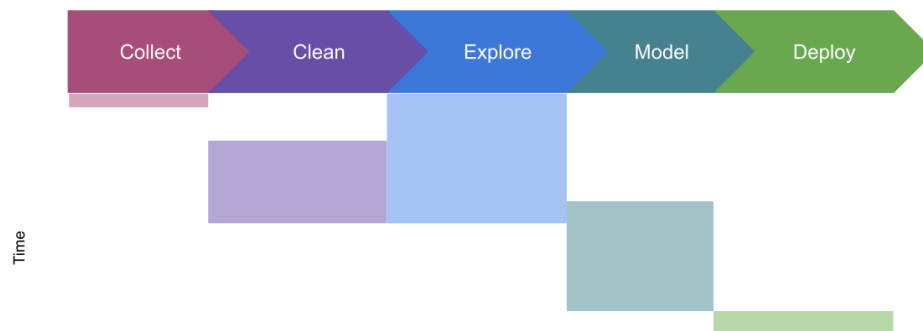


1.6.3. how we'll cover Data Science, in depth



- *collect*: Discuss only a little; Minimal programming involved
- *clean*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course
- *explore*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course
- *model*: Cover the main programming, basic idea of models; How to use models, not how learning algorithms work
- *deploy*: A little bit at the end, but a lot of preparation for decision making around deployment

1.6.4. how we'll cover it in, time



We'll cover exploratory data analysis before cleaning because those tools will help us check how we've cleaned the data.

1.7. Prepare for the next class

- Read carefully the syllabus section of the [course website](#)

- skim the rest of the [course website](#)
- Bring questions about how the class will work to class on Thursday.
- Review [Git & GitHub Fundamentals](#)
- Bring git/github questions on Thursday.
- Begin reading [chapter 1 of think like a data scientist](#) (finish in time for it to help you with the assignment due Monday night)

On Thursday we will start with a review of the syllabus. You will answer an ungraded quiz to confirm that you understand and I'll answer all of your questions. Then we will do a little bit with Git/GitHub and start your first assignment in class.

Think like a data scientist is written for practitioners; not as a text book for a class. It does not have a lot of prerequisite background, but the sections of it that I assign will help you build a better mental picture of what doing Data Science about.

Warning

Only the first assignment will be due this fast, it's a short review and setup assignment. It's due quickly so that we know that you have everything set up and the prerequisite material before we start new material next week.

Tip

In chapter 1, focus most on sections 1.1, 1.3, and 1.7.

1. Assignments

All assignments are due on Sunday at 11:59pm, via GitHub unless otherwise noted.

Assignment TOC:

Portfolio

```
/opt/hostedtoolcache/Python/3.7.15/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:26: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
/opt/hostedtoolcache/Python/3.7.15/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:31: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
```

This section of the site has a set of portfolio prompts and this page has instructions for portfolio submissions.

Starting in week 3 it is recommended that you spend some time each week working on items for your portfolio, that way when it's time to submit you only have a little bit to add before submission.

The portfolio is your only chance to earn Level 3 achievements, however, if you have not earned a level 2 for any of the skills in a given check, you could earn level 2 then instead. The prompts provide a starting point, but remember that to earn achievements, you'll be evaluated by the rubric. You can see the full rubric for all portfolios in the [syllabus](#). Your portfolio is also an opportunity to be creative, explore things, and answer your own questions that we haven't answered in class to dig deeper on the topics we're covering. Use the feedback you get on assignments to inspire your portfolio.

Each submission should include an introduction and a number of 'chapters'. The grade will be based on both that you demonstrate skills through your chapters that are inspired by the prompts and that your summary demonstrates that you *know* you learned the skills. See the [formatting tips](#) for advice on how to structure files.

On each chapter(for a file) of your portfolio, you should identify which skills by their keyword, you are applying.

You can view a (fake) example [in this repository](#) as a [pdf](#) or as a [rendered website](#)

Upcoming Checks

Portfolio 1

More information to follow

Formatting Tips

Warning

This is all based on you having accepted the portfolio assignment on github and having a cloned copy of the template. If you are not enrolled or the initial assignment has not been issued, you can view [the template on GitHub](#)

Your portfolio is a [jupyter book](#). This means a few things:

- it uses [myst markdown](#)
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

Managing Files and version

You can either convert your ipynb files to earlier to read locally or on GitHub.

The GitHub version means installing less locally, but means that after you push changes, you'll need to pull the changes that GitHub makes.

To manage with a precommit hook jupyter conversion

change your `.pre-commit-config.yaml` file to match the following:

```
repos:
-   repo: https://github.com/mwouts/jupyterx
    rev: v1.10.0 # CURRENT_TAG/COMMIT_HASH
    hooks:
    - id: jupyterx
      args: [--from, ipynb, --to, myst]
```

Run Precommit over all the files to actually apply that script to your repo.

```
pre-commit install
pre-commit run --all-files
```

If you do `git status` now, you should have a `.md` file for each `ipynb` file that was in your repository, now add and commit those.

Now, each time you commit, it will run jupyterx first.

To manage with a gh action jupyter conversion

create a file at `.github/workflows/jupyterx.yml` and paste the following:

```

name: jupytertext

# Only run this when the master branch changes
on:
  push:
    branches:
      - main
    # If your git repository has the Jupyter Book within some-subfolder next to
    # unrelated files, you can make this run only if a file within that specific
    # folder has been modified.
    #
    # paths:
    # - some-subfolder/**

# This job installs dependencies, build the book, and pushes it to `gh-pages`
jobs:
  jupytertext:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      # Install dependencies
      - name: Set up Python 3.7
        uses: actions/setup-python@v1
        with:
          python-version: 3.7

      - name: Install dependencies
        run: |
          pip install jupytertext
      - name: convert
        run: |
          jupytertext */*.ipynb --to myst
          jupytertext *.ipynb --to myst
      - uses: EndBug/add-and-commit@v4 # You can change this to use a specific version
        with:
          # The arguments for the `git add` command (see the paragraph below for more
info)
          # Default: '.'
          add: '.'

          # The name of the user that will be displayed as the author of the commit
          # Default: author of the commit that triggered the run
          author_name: Your Name

          # The email of the user that will be displayed as the author of the commit
          # Default: author of the commit that triggered the run
          author_email: you@uri.edu

          # The local path to the directory where your repository is located. You
          # should use actions/checkout first to set it up
          # Default: '.'
          cwd: '.'

          # Whether to use the --force option on `git add`, in order to bypass
eventual gitignores
          # Default: false
          force: true

          # Whether to use the --signoff option on `git commit`
          # Default: false
          signoff: true

          # The message for the commit
          # Default: 'Commit from GitHub Actions'
          message: 'convert notebooks to md'

          # Name of the branch to use, if different from the one that triggered the
workflow
          # Default: the branch that triggered the workflow (from GITHUB_REF)
          ref: 'main'

          # Name of the tag to add to the new commit (see the paragraph below for more
info)
          # Default: ''
          tag: "v1.0.0"

    env:
      # This is necessary in order to push a commit to the repo
      GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Leave this line unchanged

```

Organization

The summary of for the **part** or whole submission, should match the skills to the chapters. Which prompt you're addressing is not important, the prompts are a *starting point* not the end goal of your portfolio.

Data Files

Also note that for your portfolio to build, you will have to:

- include the data files in the repository and use a relative path OR
- load via url

using a full local path(eg that starts with `///file:`) **will not work** and will render your portfolio unreadable.

Structure of plain markdown

Use a heading like this:

```
# Heading of page
## Heading 2
### Heading 3
```

in the file and it will appear in the sidebar.

You can also make text *italic* or **bold** with either **asterics** or *__underscores__* with *_one_* for *italic_* or ***two_* for *bold*** in either case

File Naming

It is best practice to name files without spaces. Each **chapter** or file should have a descriptive file name (*with_no_spaces*) and descriptive title for it.

Syncing markdown and ipynb files

If you have the precommit hook working, git will call a script and convert your notebook files from the ipynb format (which is json like) to Myst Markdown, which is more plain text with some header information. The markdown format works better with version control, largely because it doesn't contain the outputs.

If you don't get the precommit hook working, but you do get jupyter installed, you can set each file to sync.

Adding annotations with formatting or margin notes

You can either install [jupyter](#) and convert locally or upload /push a notebook to your repository and let GitHub convert.

Then edit the .md file with a [text editor](#) of your choice. You can run by uploading if you don't have jupyter installed, or locally if you have installed jupyter or jupyterbook.

In your .md file use backticks to mark [special content blocks](#)

```
```{note}
Here is a note!
```
```

```
```{warning}
Here is a warning!
```
```

```
```{tip}
Here is a tip!
```
```

```
```${margin}
Here is a margin note!
```
```

For a complete list of options, see [the sphinx-book-theme documentation](#).

Links

Markdown syntax for links

```
[text to show](path/or/url)
```

Configurations

Things like the menus and links at the top are controlled as [settings](#), in `_config.yml`. The following are some things that you might change in your configuration file.

Show errors and continue

To show errors and continue running the rest, add the following to your configuration file:

```
# Execution settings
execute:
  allow_errors      : true
```

Using additional packages

You'll have to add any additional packages you use (beyond pandas and seaborn) to the `requirements.txt` file in your portfolio.

FAQ

This section will grow as questions are asked and new content is introduced to the site. You can submit questions:

- via e-mail to Dr. Brown ([brownsarahm](#)) or Beibhinn ([beibhinn](#))
- via [Prismia.chat](#) during class
- by creating an [issue](#)

Syllabus and Grading FAQ

How much does assignment x, class participation, or a portfolio check weigh in my grade?

There is no specific weight for any activities, because your grade is based on earning achievements for the skills listed in the [skills rubric](#).

However, if you do not submit (or earn no achievements from) assignments or portfolios, the maximum grade you can earn is a C. If you do not submit (or earn no achievements from) your portfolio, the maximum grade you can earn is a B.

Can I submit this assignment late if ...?

Late assignments are not accepted, however, your grade is based on the skills, not the assignments. All skills are assessed in at least two [assignments](#), so missing any one will not hurt your grade. If you need an accommodation because you cannot submit multiple assignments, contact Dr. Brown.

I don't understand my grade on this assignment

If you have questions about your grade, the best place to get feedback is to reply on the Feedback PR. Either reply directly to one of the inline comments, or the summary.

Be specific about what you think you should have earned and why.

Git and GitHub

I can't push to my repository, I get an error that updates were rejected

```
! [rejected] main -> main (fetch first)
error: failed to push some refs to <repository name>
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Your local version and github version are out of sync, you need to pull the changes from github to your local computer before you can push new changes there.

After you run

```
git pull
```

You'll probably have to [resolve a merge conflict](#)

The content I added to my portfolio isn't in the pdf

There was an error in the original `_toc.yml` file, change yours to match the following:

```
format: jb-book
root: intro
parts:
  - caption: About
    chapters:
      - file: about/index
      - file: about/grading
  # - caption: Check 1
  #   chapters:
  #     - file: submission_1_intro
```

uncomment the later lines and add any new files you add.

My command line says I cannot use a password

GitHub has [strong rules](#) about authentication You need to use SSH with a public/private key; HTTPS with a [Personal Access Token](#) or use the [GitHub CLI auth](#)

My .ipynb file isn't showing in the staging area or didn't push

.ipynb files are json that include all of the output, including tables as html and plots as svg, so, unlike plain code files, they don't play well with version control.

Your portfolio has `*/*.ipynb` in the `.gitignore` file, so that these files do not end up in your repository. Instead, you'll convert your notebooks to [Myst Markdown](#) with [jupytertext](#) via a [precommit hook](#).

Your portfolio has the code to do this already, what you should do is make sure that `pre-commit` is installed and then run `pre-commit install` (see your portfolio's [README.md](#) file for more detail)

If this doesn't work, you can follow the alternative in the portfolio readme.

If that doesn't work, and you have time before the deadline, create an issue to get help.

As a last resort, use the jupyter interface to download (File > Download as > ...)your notebook as `.md` if available or `.py` if not and then move that file from your Downloads folder to your repository. We'll set up another workflow for future work

My portfolio won't compile

If there's an error your notebook it can't complete running. You can allow it to run if the error is on purpose by changing settings as mentioned on the formatting page.

Help! I accidentally merged the Feedback Pull Request before my assignment was graded

That's ok. You can fix it.

You'll have to work offline and use GitHub in your browser together for this fix. The following instructions will work in terminal on Mac or Linux or in GitBash for Windows. (see Programming Environment section on the tools page).

First get the url to clone your repository (unless you already have it cloned then skip ahead): on the main page for your repository, click the green "Code" button, then copy the url that's show

The screenshot shows the GitHub repository page for `rhodyprog4ds/portfolio-brownsarahm`. At the top, there's a notification: "feedback had recent pushes 1 minute ago" with a "Compare & pull request" button. Below this, the repository is on the `main` branch with 5 branches and 1 tag. A table lists the repository files:

| File/Folder | Description |
|-----------------------------|-------------------------------------|
| <code>.github</code> | correct path for jupyter conversion |
| <code>about</code> | mvoe notebook |
| <code>template_files</code> | convert notebooks to md |
| <code>.gitignore</code> | merge gh changes and ignore |
| <code>README.md</code> | Initial commit |

On the right, the "Code" dropdown menu is open, showing options: "Clone with HTTPS" (with a "Use SSH" link), "Open with GitHub Desktop", and "Download ZIP". The HTTPS URL is `https://github.com/rhodyprog4ds/por`.

Next open a terminal or GitBash and type the following.

```
git clone
```

then past your url that you copied. It will look something like this, but the last part will be the current assignment repo and your username.

```
git clone https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
```

When you merged the Feedback pull request you advanced the `feedback` branch, so we need to hard reset it back to before you did any work. To do this, first check it out, by navigating into the folder for your repository (created when you cloned above) and then checking it out, and making sure it's up to date with the `remote` (the copy on GitHub)

```
cd portfolio-brownsarahm
git checkout feedback
git pull
```

Now, you have to figure out what commit to revert to, so go back to GitHub in your browser, and swithc to the feedback branch there. Click on where it says `main` on the top right next to the branch icon and choose feedback from the list.

rhodyprog4ds / portfolio-brownsarahm Private

generated from rhodyprog4ds/portfolio

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

🔔 feedback had recent pushes 1 minute ago [Compare & pull request](#)

🔗 main 5 branches 1 tag [Go to file](#) [Add file](#) [Code](#)

Switch branches/tags

Branches Tags
✓ main default
feedback
gh-pages
someOtherBranch

notebook ✓ a6f7f45 15 minutes ago ⌚ 14 commits
correct path for jupyter conversion 17 hours ago
mvoe notebook 17 minutes ago
convert notebooks to md 17 hours ago
merge gh changes and ignore 3 days ago
Initial commit 3 days ago

Now view the list of all of the commits to this branch, by clicking on the clock icon with a number of commits

rhodyprog4ds / portfolio-brownsarahm Private

generated from rhodyprog4ds/portfolio

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

🔔 feedback had recent pushes 15 minutes ago [Compare & pull request](#)

🔗 feedback 5 branches 1 tag [Go to file](#) [Add file](#) [Code](#)

This branch is 1 commit ahead of main. [Pull request](#) [Compare](#)

brownsarahm Merge pull request #1 from rhodyprog4ds/main f301d90 16 minutes ago ⌚ 15 commits

.github correct path for jupyter conversion 17 hours ago
about mvoe notebook 20 minutes ago
template_files convert notebooks to md 17 hours ago

On the commits page scroll down and find the commit titled “Setting up GitHub Classroom Feedback” and copy its hash, by clicking on the clipboard icon next to the short version.

more examples
brownsarahm committed 3 days ago

9427c13

convert notebooks to md
brownsarahm committed 3 days ago

e2f5b79

Update jupyter_ipynb_md.yml
brownsarahm committed 3 days ago ✓

Verified 7bd76c6

solution
brownsarahm committed 3 days ago ✓

fbc6613

Setting up GitHub Classroom Feedback
brownsarahm committed 3 days ago ✗

822cfe5

GitHub Classroom Feedback
brownsarahm committed 3 days ago ✗

f3e0297

Initial commit
brownsarahm committed 3 days ago ✓

66c21c3

[Newer](#) [Older](#)

Now, back on your terminal, type the following

```
git reset --hard
```

then paste the commit hash you copied, it will look something like the following, but your hash will be different.

```
git reset --hard 822cfe51a70d356d448bcaede5b15282838a5028
```

If it works, your terminal will say something like

```
HEAD is now at 822cfe5 Setting up GitHub Classroom Feedback
```

but the number on yours will be different.

Now your local copy of the `feedback` branch is reverted back as if you had not merged the pull request and what's left to do is to push those changes to GitHub. By default, GitHub won't let you push changes unless you have all of the changes that have been made on their side, so we have to tell Git to force GitHub to do this.

Since we're about to do something with forcing, we should first check that we're doing the right thing.

```
git status
```

and it should show something like

```
On branch feedback
Your branch is behind 'origin/feedback' by 12 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
```

Your number of commits will probably be different but the important things to see here is that it says `On branch feedback` so that you know you're not deleting the `main` copy of your work and `Your branch is behind origin/feedback` to know that reverting worked.

Now to make GitHub match your reverted local copy.

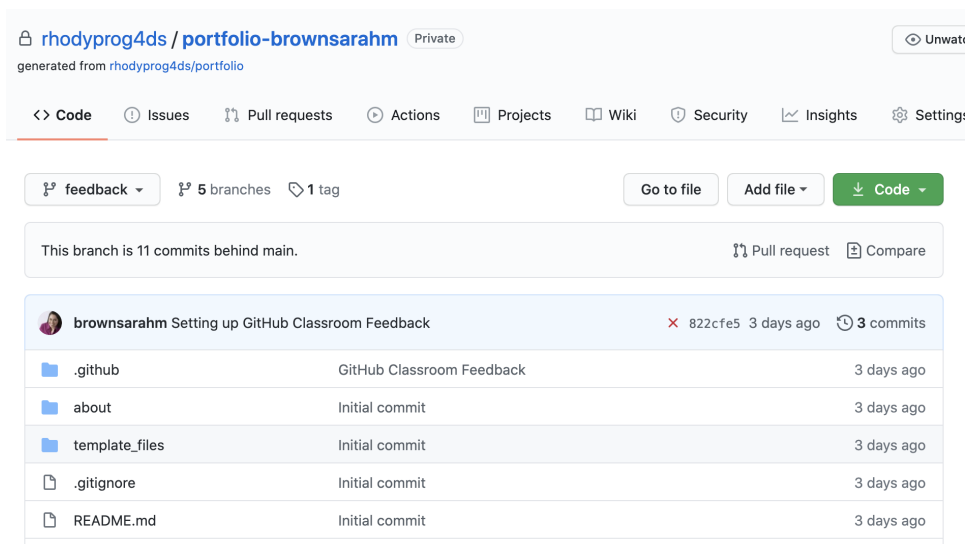
```
git push origin -f
```

and you'll get something like this to know that it worked

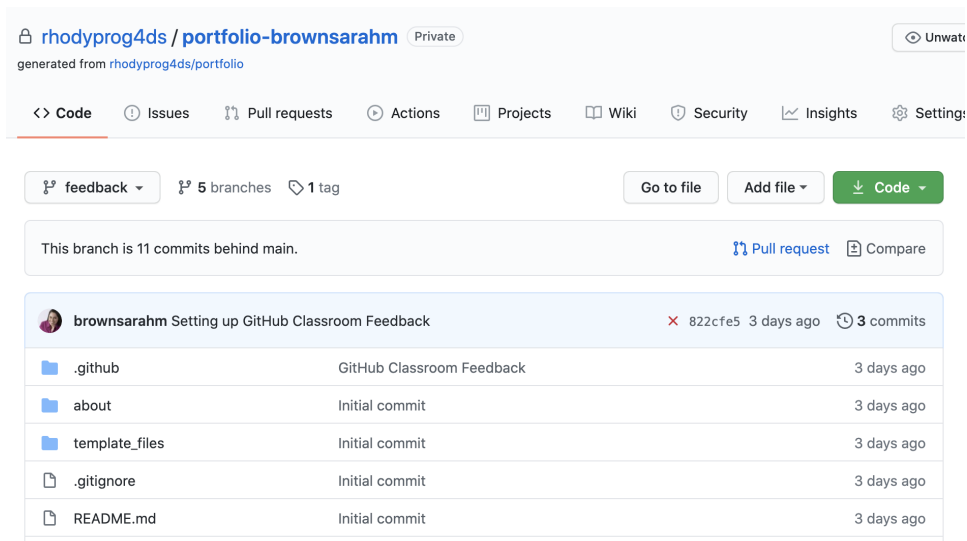
```
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
+ f301d90...822cfe5 feedback -> feedback (forced update)
```

Again, the numbers will be different and it will be your url, not mine.

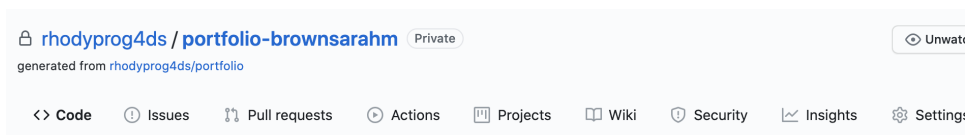
Now back on GitHub, in your browser, click on the code tab. It should look something like this now. Notice that it says, "This branch is 11 commits behind main" your number will be different but it should be 1 less than the number you had when you checked `git status`. This is because we reverted the changes you made to main (11 for me) and the 1 commit for merging main into feedback. Also the last commit (at the top, should say "Setting up GitHub Classroom Feedback").



Now, you need to recreate your Pull Request, click where it says pull request.

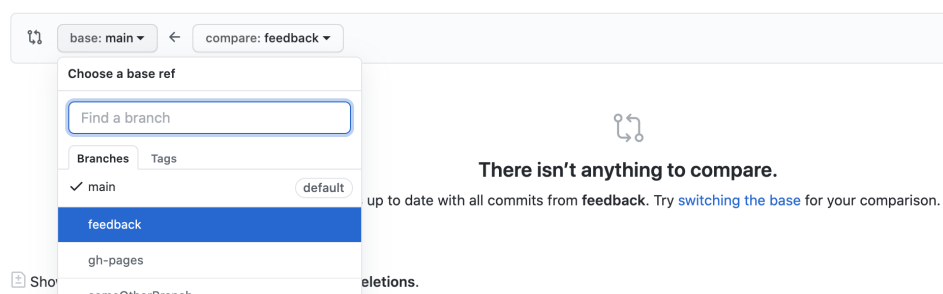


It will say there isn't anything to compare, but this is because it's trying to use **feedback** to update **main**. We want to use **main** to update **feedback** for this PR. So we have to swap them. Change base from **main** to **feedback** by clicking on it and choosing **feedback** from the list.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



Then the change the compare **feedback** on the right to **main**. Once you do that the page will change to the “Open a Pull Request” interface.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

Make the title "Feedback" put a note in the body and then click the green "Create Pull Request" button.

Now you're done!

If you have trouble, create an issue and tag [@rhodyprog4ds/fall22instructors](#) for help.

Code Errors

Key Error

If you get a key error for a pandas operation, it means that the column name as you typed it is not in the DataFrame. Check the spelling, leading or trailing whitespace can be especially troubling.

<bound method

You're probably missing `()` on a method, so Python returned the method itself as an object instead of calling it and returning the output.

Glossary

Ram Token Opportunity

Contribute glossary items and links for further reading using the suggest an edit button behind the GitHub menu at the top of the page.

[aggregate](#)

to combine data in some way, a function that can produce a customized summary table

[anonymous function](#)

a function that's defined on the fly, typically to lighten syntax or return a function within a function. In python, they're defined with the [lambda](#) keyword.

[BeautifulSoup](#)

a python library used to assist in web scraping, it pulls data from html and xml files that can be parsed in a variety of different ways using different methods.

corpus

(NLP) a set of documents for analysis

[DataFrame](#)

a data structure provided by pandas for tabular data in python.

[dictionary](#)

(data type) a mapping array that matches keys to values. (in NLP) all of the possible tokens a model knows

document

unit of text for analysis (one sample). Could be one sentence, one paragraph, or an article, depending on the goal

gh

GitHub's command line tools

git

a version control tool; it's a fully open source and always free tool, that can be hosted by anyone or used without a host, locally only.

GitHub

a hosting service for git repositories

index

(verb) to index into a data structure means to pick out specified items, for example index into a list or a index into a data frame. Indexing usually involves square brackets `[]` (noun) the index of a dataframe is like a column, but it can be used to refer to the rows. It's the list of names for the rows.

interpreter

the translator from human readable python code to something the computer can run. An interpreted language means you can work with python interactively

iterate

To do the same thing to each item in an [iterable](#) data structure, typically, an iterable type. Iterating is usually described as iterate over some data structure and typically uses the `for` keyword

[iterable](#)

any object in python that can return its members one at a time. The most common example is a list, but there are others.

kernel

in the jupyter environment, [the kernel](#) is a language specific computational engine

[lambda](#)

they keyword used to define an anonymous function; lambda functions are defined with a compact syntax `<name> = lambda <parameters>: <body>`

[PEP 8](#)

[Python Enhancement Proposal](#) 8, the Style Guide for Python Code.

repository

a project folder with tracking information in it in the form of a `.git` file

suffix

additional part of the name that gets added to end of a name in a merge operation

[Series](#)

a data structure provided by pandas for single columnar data with an index. Subsetting a Dataframe or applying a function to one will often produce a Series

[Split Apply Combine](#)

a paradigm for splitting data into groups using a column, applying some function (aggregation, transformation, or filtration) to each piece and combining the individual pieces back together to a single table

stop words

Words that do not convey important meaning, we don't need them (like a, the, an,). Note that this is context dependent. These words are removed when transforming text to numerical representation

test accuracy

percentage of predictions that the model predict correctly, based on held-out (previously unseen) test data

[Tidy Data Format](#)

Tidy data is a database format that ensures data is easy to manipulate, model and visualize. The specific rules of Tidy Data are as follows: Each variable is a column, each row is an observation, and each observable unit is a table.

token

a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing (typically a word, but more general)

TraceBack

an error message in python that traces back from the line of code that had caused the exception back through all of the functions that called other functions to reach that line. This is sometimes called tracing back through the stack

training accuracy

percentage of predictions that the model predict correctly, based on the training data

Web Scraping

the process of extracting data from a website. In the context of this class, this is usually done using the python library beautiful soup and a html parser to retrieve specific data.

References on Python

Official Documentation

- [Python](#)
- [Pandas](#)
- [Matplotlib](#)
- [Seaborn](#)

Key Resources

- [Course Text](#) this book roughly covers things that we cover in the course, but since things change quickly, we don't rely on it too closely
- [Real Python](#) this site includes high quality tutorials
- [Towards Data Science](#) this blog has some good tutorials, but old ones are not always updated, so always check the date and don't rely too much on posts more than 2 years old.



Ram Token Opportunity

If you find other high quality, reliable sources that you want to share, you can earn ram tokens.

Cheatsheet

Patterns and examples of how to use common tips in class

How to use brackets

| symbol | use |
|-------------------------------------|---|
| <code>[val]</code> | indexing item val from an object; <code>val</code> is int for iterables, or any for mapping |
| <code>[val : val2]</code> | slicing elemtns val to val2-1 from a listlike object |
| <code>[item1,item2]</code> | creating a list consisting of <code>item1</code> and <code>item2</code> |
| <code>(param)</code> | function calls |
| <code>(item1,item2)</code> | defining a tuple of <code>item1</code> and <code>item2</code> |
| <code>{item1,item2}</code> | defining a set of <code>item1</code> and <code>item2</code> |
| <code>{key:val1, key2: val2}</code> | defining a dictionary where key1 indexes to val2 |

Axes

First build a small dataset that's just enough to display

```
data = [[1,0],[5,4],[1,4]]
df = pd.DataFrame(data = data,
                  columns = ['A', 'B'])
df
```

| | A | B |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 5 | 4 |
| 2 | 1 | 4 |

This data frame is originally 3 rows, 2 columns. So summing across rows will give us a [Series](#) of length 3 (one per row) and long columns will give length 2, (one per column). Setting up our toy dataset to not be a square was important so that we can use it to check which way is which.

```
df.sum(axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.sum(axis=1)
```

```
0    1
1    9
2    5
dtype: int64
```

```
df.apply(sum,axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.apply(sum,axis=1)
```

```
0    1
1    9
2    5
dtype: int64
```

Indexing

```
df['A'][1]
```

```
5
```

```
df.iloc[0][1]
```

```
0
```

Data Sources

This page is a semi-curated source of datasets for use in assignments. The different sections have datasets that are good for different assignments.

Best for loading directly into a notebook

- [Tidy Tuesday](#) inside the folder for each year there is a README file with list of the datasets. These are .csv files
- [Json Datasets](#)
- [National Center for Education Statistics Digest 2019](#) These data tables are available for download as excel and visible on the page.
- Lots of wikipedia pages have tables in them.

General Sources

These may require some more work

- [Stackoverflow Developer Survey](#) This data comes with readme info all packaged together in a .zip. You'll need to unzip it first.
- [Google Dataset Search](#)
- [Kaggle](#) most Kaggle datasets will require you to download and unzip them first and then you can copy them into your repo folder.
- [UCI Data Repository](#) Machine Learning focused datasets, can filter by task
- [A curated list of datasets by task](#) It includes datasets for cleaning, visualization, machine learning, and "data analysis" which would align with EDA in this course.
- [Hugging Face NLP Datasets](#) lots of text datasets

Datasets in many parts

- [Makeup Shades](#)
- [Kenya Census](#)
- [Wealth and Income over time](#)
- [UN Votes](#)
- [Deforestation](#)
- [Survivor](#)
- [Billboard](#)
- [Caribou Tracking](#)
- [Video games from steam 2021](#) and from [2019](#)
- [BBC Rap Artists](#)

Datasets with time

- [Superbowl commercials](#)

Databases

- [SQLite Databases](#)

If you have others please share by creating a pull request or issue on this repo (from the GitHub logo at the top right, [suggest](#) [edit](#)).

General Tips and Resources

This section is for materials that are not specific to this course, but are likely useful. They are not generally required readings or installs, but are options or advice I provide frequently.

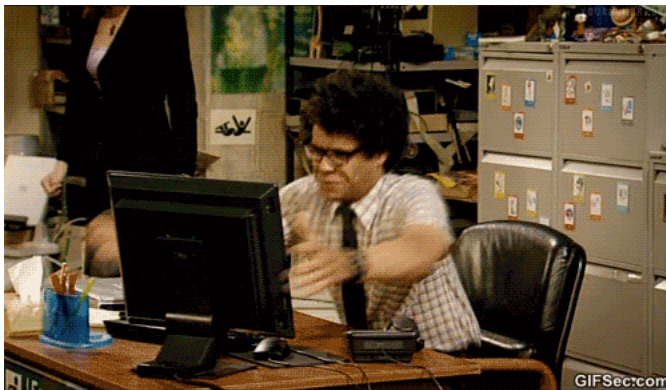
on email

- [how to e-mail professors](#)

How to Study in this class

This is a programming intensive course and it's about data science. This course is designed to help you learn how to program for data science and in the process build general skills in both programming and using data to understand the world. Learning two things at once is more complex. In this page, I break down how I expect learning to work for this class.

Remember the goal is to avoid this:



Why this way?

Learning to program requires iterative practice. It does not require memorizing all of the specific commands, but instead learning the basic patterns.

Using reference materials frequently is a built in part of programming, most languages have built in help as a part of the language for this reason. This course is designed to have you not only learn the material, but also to build skill in learning to program. Following these guidelines will help you build habits to not only be successful in this class, but also in future programming.

A new book that might be of interest if you find programming classes hard is [the Programmers Brain](#) As of 2021-09-07, it is available for free by clicking on chapters at that linked table of contents section.

i Where are your help tools?

In Python and Jupyter notebooks, what help tools do you have?

Learning in class

Important

My goal is to use class time so that you can be successful with *minimal frustration* while working outside of class time.

Programming requires both practical skills and abstract concepts. During class time, we will cover the practical aspects and introduce the basic concepts. You will get to see the basic practical details and real examples of debugging during class sessions. Learning to debug something you've never encountered before and setting up your programming environment, for example, are *high frustration* activities, when you're learning, because you don't know what you don't know. On the other hand, diving deeper into options and more complex applications of what you have already seen in class, while challenging, is something I'm confident that you can all be successful at with minimal frustration once you've seen basic ideas in class. My goal is that you can repeat the patterns and processes we use in class outside of class to complete assignments, while acknowledging that you will definitely have to look things up and read documentation outside of class.

Each class will open with some time to review what was covered in the last session before adding new material.

To get the most out of class sessions, you should have a laptop with you. During class you should be following along with Dr. Brown, typing and running the same code. You'll answer questions on Prismia chat, when you do so, you should try running necessary code to answer those questions. If you encounter errors, share them via prismia chat so that we can see and help you.

After class

After class, you should practice with the concepts introduced.

This means reviewing the notes: both yours from class and the annotated notes posted to the course website.

When you review the notes, you should be adding comments on tricky aspects of the code and narrative text between code blocks in markdown cells. While you review your notes and the annotated course notes, you should also read the documentation for new modules, libraries, or functions introduced that day.

In the annotated notes, there will often be extra questions or ideas on how to extend and practice the concepts. Try these out.

If you find anything hard to understand or unclear, write it down to bring to class the next day.

Assignments

In assignments, you will be asked to practice with specific concepts at an intermediate level.

Assignments will apply the concepts from class with minimal extensions. You will probably need to use help functions and read documentation to complete assignments, but mostly to look up things you saw in class and make minor variations. Most of what you need for assignments will be in the class notes, which is another reason to read them after class.

Portfolios

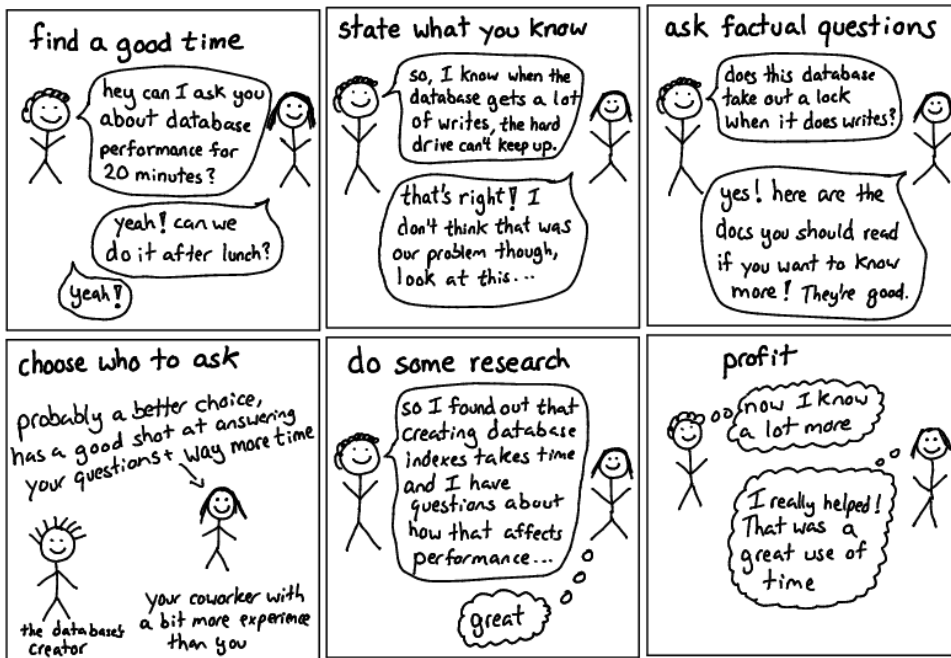
In portfolios, your goal is to extend and apply the concepts taught in class and practiced in assignments to solve more realistic problems. You may also reflect on your learning in order to demonstrate deep understanding. These will require significant reading beyond what we cover in class.

Getting Help with Programming

Asking Questions

JULIA EVANS
@b0rk

asking good questions



One of my favorite resources that describes how to ask good questions is [this blog post](#) by Julia Evans, a developer who writes comics about the things she learns in the course of her work and publisher of [wizard zines](#).

Describing what you have so far

Stackoverflow is a common place for programmers to post and answer questions.

As such, they have written a good [guide on creating a minimal, reproducible example](#).

Creating a minimal reproducible example may even help you debug your own code, but if it does not, it will definitely make it easier for another person to understand what you have, what your goal is, and what's working.

Note

A fun version of this is [rubber duck debugging](#)

Understanding Errors

Error messages from the compiler are not always straight forward.

The [TraceBack](#) can be a really long list of errors that seem like they are not even from your code. It will trace back to all of the places that the error occurred. It is often about how you called the functions from a library, but the compiler cannot tell that.

To understand what the traceback is, how to read one, and common examples, see [this post on Real Python](#).

One thing to try, is [friendly traceback](#) a python package that is designed to make that error message text more clear and help you figure out what to do next.

Ram Token Opportunity

If you try out friendly traceback and find it helpful, add a testimonial here. using

```
```{epigraph}
```

## Terminals and Environments

## Why all this work?

Managing environments is **one of the hardest parts of programming** so, as instructors, we often design our courses around not having to do it. In this class, however, I'm choosing to take the risk and help you all through beginning to manage your own environments.

These issues will be the most painful in the course, I promise.

I think it's worth this type of pain though, because all the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your independence as a programmer. This also means responsibility and some messy debugging, but I think this is a good tradeoff. This is an upper level (300+) level course, so increasing some complexity is expected and I want as much as possible to keep you close to realistic programming environments; so that what you see in this course is **directly, and immediately**, applicable in real world contexts. You should be able to pick up data science side projects or an internship with ease after this course.

I *know* some of these things will be frustrating at times, but I want you to feel supported in that and know that your grade will not be blocked by you having environment issues, as long as you ask for help in a timely matter.

## Windows

Windows has a sort of multiverse of terminal environments.

The least setup required involves using anaconda prompt and `conda` to manage your python environment and GitBash to work with git (and it can also do other bash related things).

Instead of managing two terminals, you may [configure your path in GitBash to make Anaconda work](#)

## MacOS

MacOS has one terminal app, but it can run different shells.

On MacOS You may want to switch to bash (using the `bash` command or make it your default and [update bash](#)).

## Getting Organized for class

The only **required** things are in the Tools section of the syllabus, but this organizational structure will help keep you on top of what is going on.

Your username will be appended to the end of the repository name for each of your assignments in class.

## File structure

I recommend the following organization structure for the course:

```
CSC310
|- notes
|- portfolio-username
|- 02-accessing-data-username
|- ...
```

This is one top level folder with all materials in it. A folder inside that for in class notes, and one folder per repository.

Please **do not** include all of your notes or your other assignments all inside your portfolio, it will make it harder to grade.

## Finding repositories on github

### Note

We know that we don't currently teach a lot of this in our department, so in Spring 22 I'm teaching a brand new course on Computer Systems, that will help you understand the underlying concepts that make all of this stuff make sense, instead of just following recipes and debugging here and there.

If, for example, you come to me in week 5 and have never got an any environment working and you're trying for the first time, your grade will be hurt because you will be very far behind at that point. Ask for help early and often.

Each assignment repository will be created on GitHub with the [rhodyprog4ds](#) organization as the owner, not your personal account. Since your account is not the owner, they do not show on your profile.

Your assignment repositories are all private during the semester. At the end, you may take ownership of your portfolio[^pttrans] if you would like.

If you go to the main page of the [organization](#) you can search by your username (or the first few characters of it) and see only your repositories.

#### Warning

Don't try to work on a repository that does not end in your username; those are the template repositories for the course and you don't have edit permission on them.

## Letters to Future students

This section is a place for students enrolled in Fall 2020 to write letters to future students taking this class with Professor Brown. The websites for future sections will link back here for them to read.

## Contributed Notes

### Reviewing notes

Especially when it comes to the difficult topics make sure you go back through the notes and try and add to them yourself. Actively using the new topics will help you learn a lot better than just copying the notes.

### Attend Office Hours

Attending office hours will help you better understand course material, complete homework assignments, and learn new things that might not normally come up in class.

- David

## To contribute

Via GitHub directly:

1. Use the edit button above to add a note to this file following the example that's commented out
2. create a pull request