

# About this Book

## Contents

### Syllabus

- About
- Tools and Resources
- Data Science Achievements
- Grading
- Grading Policies
- Course Style Guide
- Support
- General URI Policies
- Communications & Office Hours

### Notes

- 1. Welcome & What is Data Science

### Assignments

- 1. Assignment 1: Setup, Syllabus, and Review

### Portfolio

- Earning Level 3
- Formatting Tips
- Generic Extension Ideas
- Alternatives to Extending Assignments for Level 3
- Process Level 3

### FAQ

- FAQ
- Syllabus and Grading FAQ
- Git and GitHub
- Code Errors

### Resources

- Glossary
- References on Python
- Cheatsheet
- Data Sources

[Skip to main content](#)

---

- 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

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

This class meets TTh 3:30-4:45 in Pastore 259.

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

## Land University of Rhode Island Land Acknowledgment

### Note

The University of Rhode Island land acknowledgment is a statement written by members of the University community in close partnership with members of the Narragansett Tribe. The statement recognizes and pays tribute to the people who lived on and stewarded the land on which the University now resides. The statement seeks to show gratitude and respect to Indigenous people and cultures and build community with the Narragansett Nation and other Native American tribes.

The University of Rhode Island occupies the traditional stomping ground of the Narragansett Nation and the Niantic People. We honor and respect the enduring and continuing relationship between the Indigenous people and this land by teaching and learning more about their history and present-day communities, and by becoming stewards of the land we, too, inhabit.

## 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

[Skip to main content](#)

### 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

### Check your Comprehension

Questions to use to check your comprehension will looklike 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.

[Skip to main content](#)

## 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 by the end of the semester.

## About this semester

This semester, I will be trying some new ways to update the course to reflect the reality that in a job, a lot of your work may be done with an AI assistant to help. That said, *learning* the basic material still has to happen, you cannot supervise an AI if you do not know what correct looks like.

Each assignment will have specific AI use guidelines that you must follow in addition to general overall course style guide and requirements.

Additionally, the grading that we will use this semester will be new, if something does not make sense, ask! I will never change a policy in a way that could hurt a student who was acting in good faith (if you were trying to game things, I may close loopholes in ways that do not benefit you. )

## About this syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly, and online at [rhodyprog4ds.github.io/BrownFall24/syllabus](https://rhodyprog4ds.github.io/BrownFall24/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 on GitHub Org Page

[Skip to main content](#)

---

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.

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 links to 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

[Skip to main content](#)

[rhodyprog4ds.github.io/BrownSpring23/](https://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 use the [GitHub CLI](#) or [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

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.

### 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
```

[Skip to main content](#)

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](#)

[Skip to main content](#)

[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 in . 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.

Each Assignment will have a deadline posted on the assignment page, typically the same day each week. Portfolio deadlines will be announced at least 2 weeks in advance.



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]

## Achievement Definitions

The table below describes how your work 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.

		skill	Level 1	Level 2	Level 3
keyword					
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 standard 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 and compute 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
optimize	Optimize model parameters		Identify when model parameters need to be	Optimize basic model parameters such as	Select optimal parameters based of multiple quantitative criteria and automate parameter tuning

[Skip to main content](#)

keyword	skill	Level 1	Level 2	Level 3
<b>compare</b>	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
<b>representation</b>	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
<b>workflow</b>	use industry standard data science tools and workflows to solve data science problems	Solve well structured fully specified problems with a single tool pipeline	Solve well-structured, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn related tools and describe strengths and weaknesses 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
<b>python</b>	1	1	0	1	1	0	0	0	0	0	0	0	0	4
<b>process</b>	1	0	0	0	0	1	1	1	1	1	1	0	0	7
<b>access</b>	0	1	1	1	1	0	0	0	0	0	0	0	0	4
<b>construct</b>	0	0	0	0	1	0	1	1	0	0	0	0	0	3
<b>summarize</b>	0	0	1	1	1	1	1	1	1	1	1	1	1	11
<b>visualize</b>	0	0	1	1	0	1	1	1	1	1	1	1	1	10
<b>prepare</b>	0	0	0	1	1	0	0	0	0	0	0	0	0	2
<b>evaluate</b>	0	0	0	0	0	1	1	1	0	1	1	0	0	5
<b>classification</b>	0	0	0	0	0	0	1	0	0	1	0	0	0	2
<b>regression</b>	0	0	0	0	0	0	0	1	0	0	1	0	0	2
<b>clustering</b>	0	0	0	0	0	0	0	0	1	0	1	0	0	2
<b>optimize</b>	0	0	0	0	0	0	0	0	0	1	1	0	0	2
<b>compare</b>	0	0	0	0	0	0	0	0	0	0	1	0	1	2
<b>representation</b>	0	0	0	0	0	0	0	0	0	0	0	1	1	2
<b>workflow</b>	0	0	0	0	0	0	0	0	0	1	1	1	1	4

[Skip to main content](#)

### Warning

**process** achievements are accumulated a little slower; details will follow.

## Extensions

### Warning

this rolling deadline is new for Fall 2024 and aims to let students distribute work in a better way for yourself. After A2 feedback is posted, I will give more explanation about how to do this, in concrete terms.

There are no extensions applicable to assignment 1, but starting after assignment 2's feedback you can start workign on level 3 achievements. You can add on and extend each analysis, once you have earned level 2 for a skill to earn level 3. You can also add new analyses that instead combine different sets of skills.

Extensions will all be graded by Dr. Brown (and most assignments will be graded by the TA Surbhi). You will make separate PRs for your attempts at level 3 from level 2.

While assignments have fixed grades, you can submit extensions as you complete them. I recommend planning to work on them consistently throughout the semester.

### Warning

In previous semesters, there were checklists, but they are removed because they distracted students from learning the important things

## 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

[Skip to main content](#)

ccessible 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

### Warning

This is going to change; you will get a notification when it is final

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, first getting the basic idea, then applying it, then exploring advanced usage. You will have to demonstrate each skill area on 3 separate occasions to get level 3 credit for it. 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.

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 learn level 1 in any assignment (even without completing it completely correct) or in office hours.
- 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. Weekly assignments will guide you to level 2, if you complete it well.
- 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. This will happen mostly extending previous assignments in your portfolio.

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 is approximately the level of understanding required for level 1.

We will not directly add these to your grade, but you can always visit office hours to earn level 1, so that your assignment can go straight to 2 in most skills.

## Assignments

For your learning to progress and earn level 2 achievements, you must practice with the skills outside of class time. You will submit all of your assignments in your portfolio repository. Sometimes you will need to sync your repo to get templates for the assignment and some will be open. All will be posted on this site.

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. Your feedback will recap all of your earned achievements to that point, not only what was earned in that assignment.

You *can* revise assignments if you do not earn achievements by also adding reflections to them while you edit, but since each skill is available in multiple assignments you do not have to.

You can revise what you submitted and resubmit it, with reflections and explanation of what you were confused about, what you tried initially, how you eventually figured it out, and explains the correct answer.

## Extensions

### Warning

the logistics of this are changing, but tbd, will update soon

To earn level 3 achievements, you will extend your prior work. Starting with assignment 2, there will be extension ideas in the assignment.

## TLDR

You *could* earn a C through oral exams in office hours alone. To earn a B, you must complete assignments. To earn an A you must complete assignments and add extensions that demonstrate deeper understanding (tips will be provided).

## Detailed mechanics

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			
<b>A</b>	15	15	15
<b>A-</b>	10	15	15
<b>B+</b>	5	15	15
<b>B</b>	0	15	15
<b>B-</b>	0	10	15
<b>C+</b>	0	5	15
<b>C</b>	0	0	15
<b>C-</b>	0	0	10
<b>D+</b>	0	0	5
<b>D</b>	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

## Grading Examples

### Getting an A Without Perfection

#### Warning

achievements are no longer awarded in class in fall 24; images to be updated and portfolio is rolling instead of discrete check points.

# 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			

### Activity Legend

In class



Assignment



Portfolio Check



### Not submitted

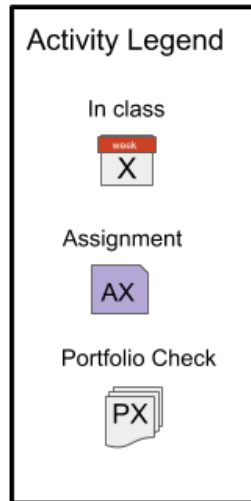


In this example, the student earned all level 1 achievements in class and all level 2 on assignments. This student was content with getting a B and chose to not submit a portfolio.

## Getting a B while having trouble

## Map to a B, having trouble

	Level 1	Level 2	Level 3
python	A1	P1	
process	A1	P2	
access	A2	P1	
construct	A5	P1	
summarize	A3	P1	
visualize	A3	P2	
prepare	A5	P2	
classification	A10	P3	
regression	A11	P2	
clustering	A9	P3	
evaluate	A11	P3	
optimize	A11	P4	
compare	A13	P3	
unstructured	A13	P4	
tools	A13	P3	



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

### Attendance

Attendance and active participation is expected. You earn level 1 achievements in class and all class sessions are active learning.

If you miss class, you can make it up by reading the posted notes and the prismia transcript. Best practice is to download them as a notebook and run them to make sure you understand each step. If you miss both class sessions in a week, the level one achievements can be made up through annotation or in your assignment.

Absences do not require notification.

### Assignment Deadlines and Late Work

**Late assignments will not be graded.** Extensions will not be granted for assignments. Every skill will be assessed through more than one assignment, so missing assignments occasionally will not necessarily impact 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

[Skip to main content](#)

If you submit work that is **not complete**, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could even be enough to earn a level 1 achievement. 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.

#### Important

If you have a serious issue during the semester, that prevents you from submitting an assignment, email Dr. Brown to make a plan. Extensions will still not be granted because they do not help you in the long run, instead an alternate plan of how to earn the target grade.

## Portfolio Deadlines and Extensions

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 at least **24 hours** before the deadline.

In this issue, include:

1. A proposed new deadline
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 assignments and portfolio check on time.
5. (if less than 24 hours before the deadline) why you need an emergency request

#### Important

Your request should not include a reason why you are asking, unless you are asking for an emergency extension. Emergency requests can be submitted at any time, even after the deadline.

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.

## Academic Dishonesty

All work must represent your own understanding of both the data science practices and the related programming concepts. Submitting code or prose that was generated by a generative model or another person is not allowed.

If you are found to have submitted work that does not constitute your own work, the following penalties apply:

- in a portfolio, all achievements attempted in the dishonest component are permanently ineligible.
- in an assignment the level three achievements for the skills of focus in the assignment are ineligible, and the relevant level

[Skip to main content](#)

For example, if you violate the academic honesty policy in assignment 4, Prepare level 3 becomes ineligible and you must meet the requirements for prepare level 3 in a portfolio in order to earn prepare level 2.

If you violate academic honesty policy in portfolio 1 while attempting level 3 at Python, access, prepare, summarize and visualize and process level 2, then your maximum grade becomes a B+, because level 3 in all five of those skills becomes ineligible.

## Regrading

1. Add comments:
  - For general questions, post on the conversation tab of your Feedback PR with your request.
  - For specific questions, reply to a specific comment.
2. Re-request a review from Dr. Brown on your Feedback Pull request.

If you think we missed *where* you did something, add a comment on that line to help us find it (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

## Course Style Guide

Following a style guide is a common requirement in companies to make it so that code written by different people stays easy to read for everyone. Consistent style also makes it easier to onboard new developers join a project and contribute faster.

The following style guide serves as practice for you following a style guide, makes your work easier to read for grading purposes, and holds you accountable to learning deeply and demonstrating that you have learned well.

## Hard Requirements

### Warning

All work must adhere to these requirements or it may receive no feedback or credit. Minor misses may receive warnings, but if submitted work does not appear to represent a good faith effort at adhering to this style guide, the only comment will be, "Follow the style guide on the next assignment"

1. All code must be submitted in a notebook file (.ipynb or myst)
2. Code must run or have explicit questions and comments about what was done about the errors
3. Python comments ( `# comment text` ) inside code cells should **only** be used to explain complex code that is not explained in the course notes. Using such code requires a citation for the source.
4. Each code cell should exactly one conceptually complete step in terms of the analysis.
5. Nearly all code cells should have output in some form
6. Every code cell must be motivated by text in markdown before it
7. Every code cell's output must be interpreted in a markdown cell (may be combined in one cell that explains the above

[Skip to main content](#)

8. the `print` function can only be used when it improves the readability over using jupyter's display, must be justified
9. No deprecated or dangerous code constructs without justification
10. All assignment questions must be answered in markdown cells
11. Notebook files may not have extraneous metadata in them
12. Alternative libraries not taught in class can only be used when attempting level 3.

## Additional Style

### ! Important

Mistakes on these will get detailed feedback once and a "see previous feedback" a second time before the whole assignment receives no feedback.

1. Code should adhere to PEP8
2. Markdown syntax should be used to enhance the readability of the text (eg not all headings, bullets where they make sense)
3. Best practices that are highlighted in class should be followed (this list will expand over the semester)

## Support

### ⚠ Warning

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

## 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. 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, 1:1 sessions with a peer tutor or a professional academic consultant. Consultations are available for individual academic

[Skip to main content](#)

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](mailto: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](http://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](http://www.uri.edu/brt). There you will also find people and resources to help.

## Mental Health and Wellness

We understand that college comes with challenges and stress associated with your courses, job/family responsibilities and personal life. URI offers students a range of services to support your [mental health and wellbeing](#), including the URI Counseling Center, MySSP (Student Support Program) App, the Wellness Resource Center, and Well-being Coaching.

## 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](http://web.uri.edu/disability), or emailing: [dss@etal.uri.edu](mailto: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


[Skip to main content](#)

k 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 directly or indirectly through the use of generative AI
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

## Communications & Office Hours

 **Warning**

Due to Dept Seminar Office hours on 11/10 will be at 5pm instead of 4pm.

## 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 notificaiton [from the notification settings page](#)

## Help Hours

Day	Time	Location	Host
TBA	TBA	Zoom	Dr. Brown
TBA	TBA	134 Tyler	Dr. Brown
TBA	TBA	TBA	Surbhi

Zoom links are on the [course organization page of GitHub](#)

## To reach out, By usage

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

	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.

## 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.

[Skip to main content](#)

 Note

Wh  
not



# 1. Welcome & What is Data Science

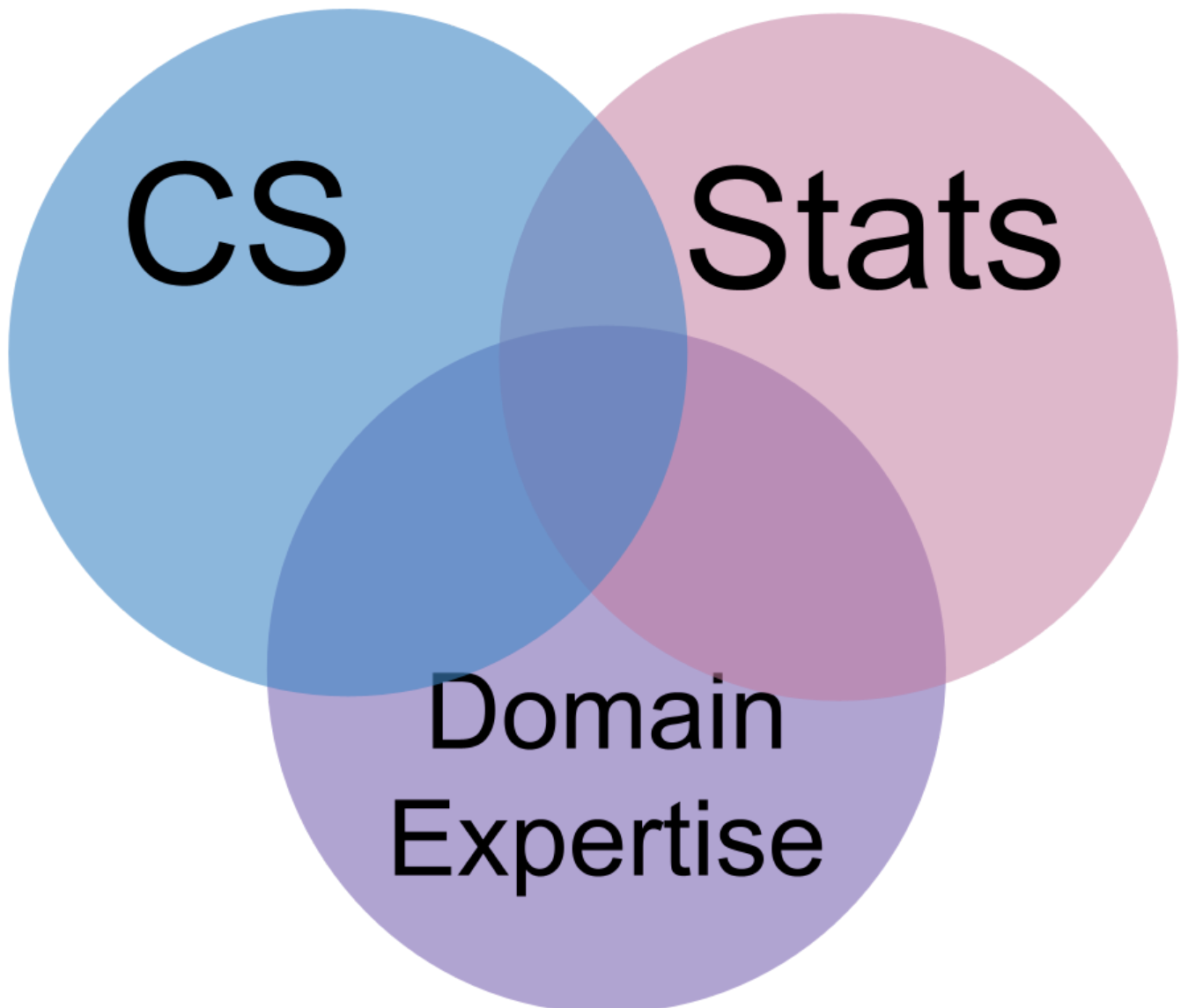
## 1.1. Prismia Chat

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. What is Data Science?

Data Science is the combination of

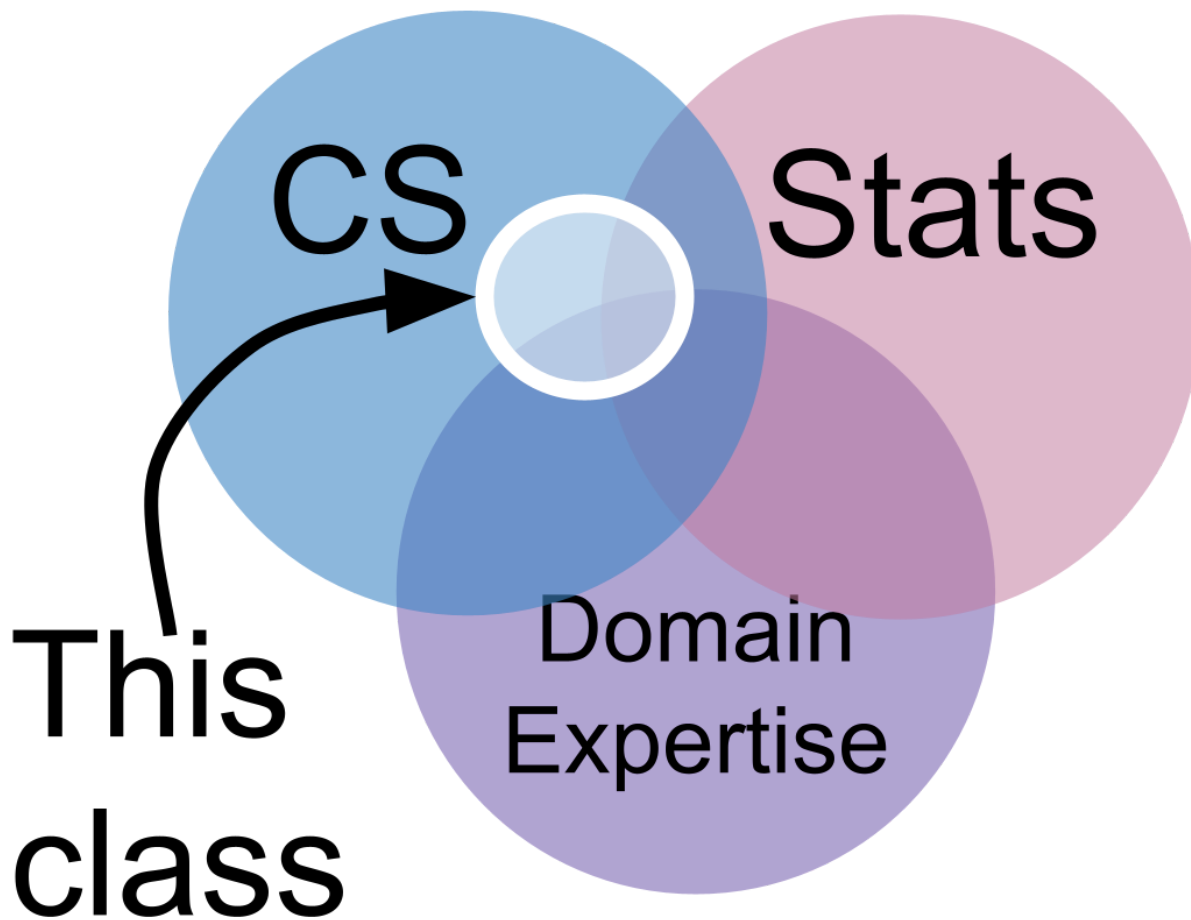


**statistics** is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

**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.

### 1.2.1. In this class,

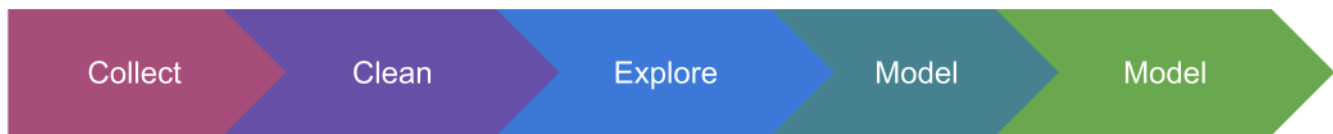


We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

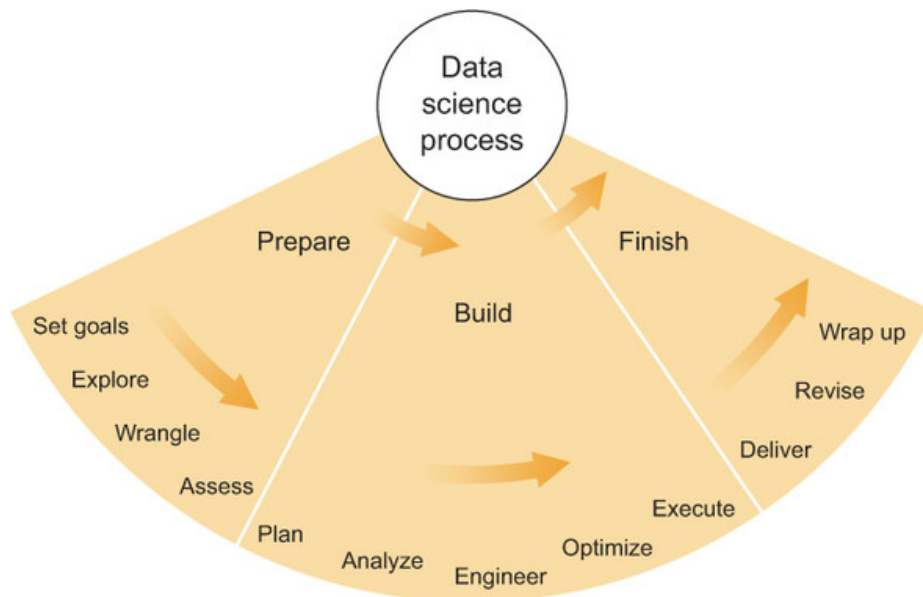
But there are many definitions. We'll use this one, but you may come across others.

### 1.2.2. How does data science happen?

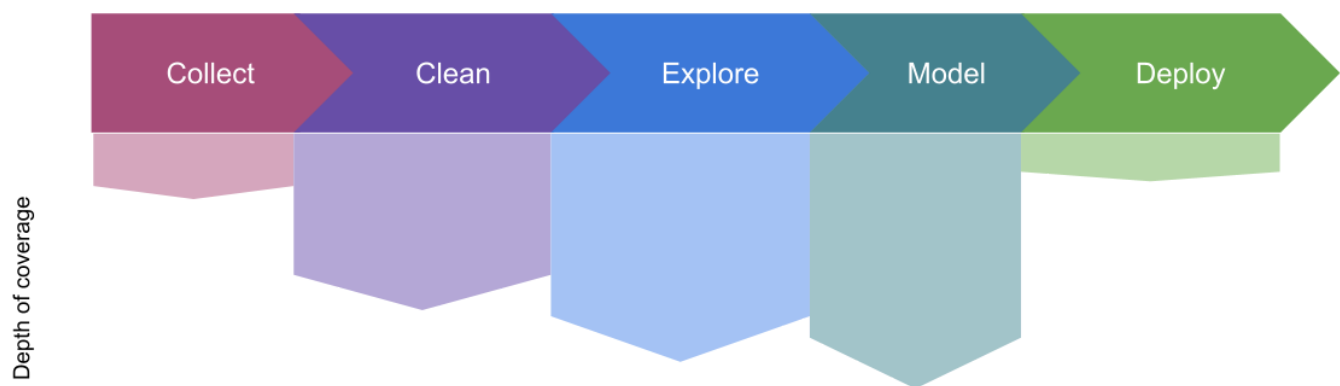
The most common way to think about what doing data science means is to think of this pipeline. It is in the perspective of the data, these are all of the things that happen to the data.



Another way to think about it



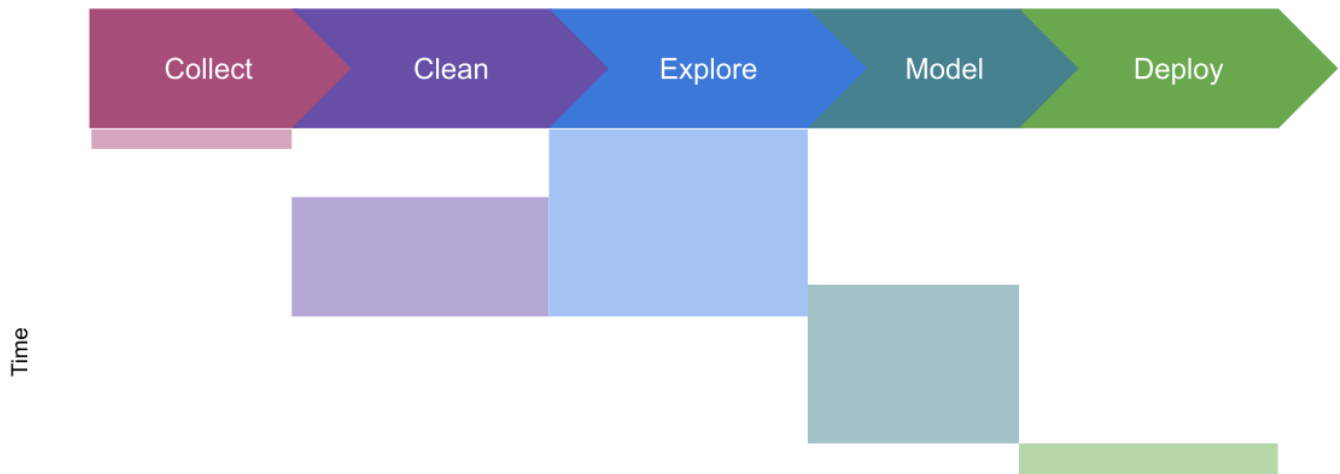
### 1.2.3. how we'll cover Data Science, in depth



- *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

[Skip to main content](#)

### 1.2.4. 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.

## 1.3. How this class will work

### Participatory Live Coding

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

#### Note

Here I changed this title from the title of the notebook, to a subsection but the rest of this is what was actually done in class, with explanation.

## 1.4. Intro to Jupyter Notebooks

## 1.5. 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.

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.

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

[Skip to main content](#)

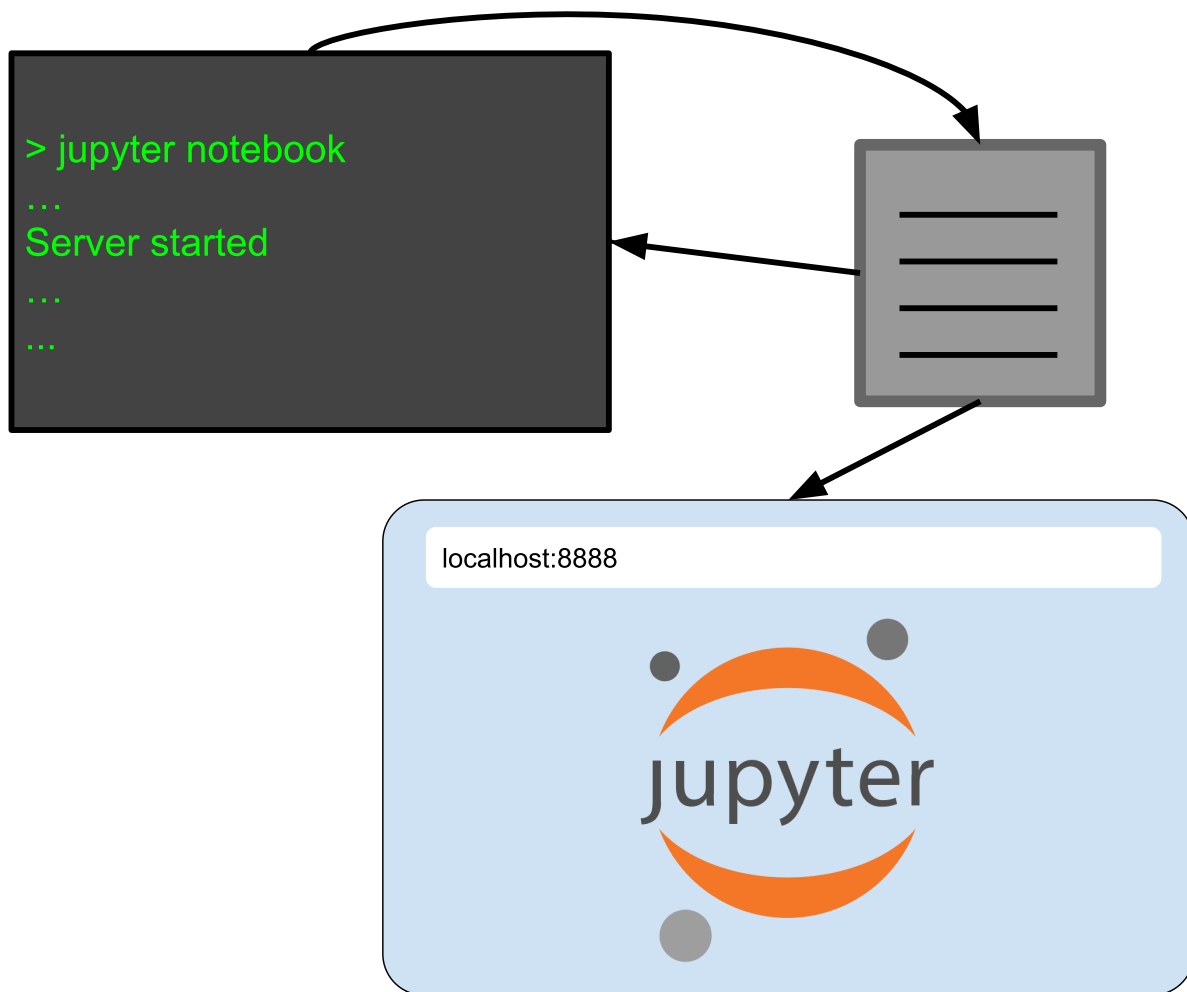
## 1.6. Jupyter Lab and Jupyter notebooks

Launch a `jupyter lab` server:

- on Windows, use anaconda terminal
- on Mac/Linux, use terminal
- `cd path/to/where/you/save/notes`
- enter `jupyter lab`

### 1.6.1. What just happened?

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



## 1.6.2. A jupyter notebook tour

A Jupyter notebook has two modes. When you first open, it is in command mode. It says the mode in the bottom right of the screen. Each box is a cell, the highlighted cell is gray when 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.

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

```
3+4
```

```
7
```

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

We can set variables

```
name = 'Sarah'
```

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

we can put a variable there to see it

```
name
```

```
'Sarah'
```

```
name_list = ['Sarah', 'Adam', 'Alex']
```

```
name_list
```

```
['Sarah', 'Adam', 'Alex']
```

#### Note

built in functions turn green in jupyter

```
print(name_list)
```

```
['Sarah', 'Adam', 'Alex']
```

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

## 1.7. 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.

```
year = '2020'
```

```
print(name, year)
```

```
Sarah 2020
```

### Tip

This is an added tip for the notes that I did not show in class, it's less useful while working, but is helpful for the notes

We can also use the `help` function

```
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.
```

The first line says that it can take multiple values, because it says `args*`. The `*` means multiple.

It also has a keyword argument (must be used like `argument=value` and has a default) described as `sep=' '`. This means that by default it adds a space as above.

The help also tells us about other parameters, like the `sep` one

```
print(name, year, 'sdkjfdsk', 'ksdjfosidj', sep='\n')
```

```
Sarah  
2020  
sdkjfdsk  
ksdjfosidj
```

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

### Important

More extra tips to help you review/refresh your Python that we did not cover in class. In general, I will not require things that we do not at least get very close to in assignments, but since this is supposed to be in the prerequisite, I am providing resources.

## 1.8. Python Review

Official source on python:

- [nen8 official style](#)

[Skip to main content](#)



We will go quickly through these focusing on pythonic style, because the prerequisite is a programming course.

## 1.9. Functions

```
def greeting(name):  
    '''  
    say hi to a person  
  
    Parameters  
    -----  
    name : string  
        the name of the person to greet  
    '''  
    return 'hi ' + name
```

A few things to note:

- the `def` keyword starts a function
- then the name of the function
- parameters in `()` then `:`
- the body is indented
- the first thing in the body should be a docstring, denoted in `'''` which is a multiline comment
- returning is more reliable than printing in a function

In python, [PEP 257](#) says how to write a docstring, but it is very broad.

In Data Science, [numpydoc](#) style docstrings are popular.

- [Pandas follows numpydoc](#)
- [\[Numpy uses it\]](#)
- [Scipy follows numpydoc](#)

Once the cell with the function definition is run, we can use the function

```
greeting(name)
```

```
'hi Sarah'
```

```
print(greeting('surbhi'))
```

```
hi surbhi
```

```
assert greeting('sarah') == 'hi sarah'
```

With a return this works to check that it does the right thing.

[Skip to main content](#)

when assert is true, it returns nothing, it throws an error on failure

## 1.10. Conditionals

```
def greeting2(name, formal=False):  
    '''  
    say hi to a person  
  
    Parameters  
    ++++++  
    name : string  
        the name of the person to greet  
    formal: bool  
        if the greeting should formal (hello) or not (hi)  
    '''  
    if formal:  
        message = 'hello ' + name  
    else:  
        message = 'hi ' + name  
    return message
```

key points in this function:

- an `if` also has the conditional part indented
- for a `bool` variable we can just use the variable
- we can set a default value

because of the default value we do not have to pass the second variable:

```
greeting2(name)
```

```
'hi Sarah'
```

```
greeting2(name, True)
```

```
'hello Sarah'
```

## 1.11. More Reading

Reading [chapter 1 of think like a data scientist](#) will help you with the data science definition part of the assignment.

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.

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.

[Skip to main content](#)

# 1. Assignment 1: Setup, Syllabus, and Review

Due: 2024-09-10 2:00pm

## ⚠ Warning

You must *complete* it by 2pm on Tuesday, but if you are confused on anything from the syllabus put `question: <your question here>`, replacing the `<>` part with our actual question in that section and then ask in class. There will be time in class to make revisions to your work before it is officially graded.

I will be reading everything before class (and I can use GitHub timestamps to see what was done before and later)

## 1.1. Evaluation

Eligible skills:

- Python
- Process

## 1.2. Related notes

- [Welcome & What is Data Science](#)

## 1.3. Instructions

### 📌 Important

If you have trouble, check the GitHub FAQ on the left first

Your task is to:

1. Install required software from the Tools & Resource page (should have been done before the first class)
2. Create your portfolio, using the link on Brightspace
3. Learn about your portfolio from the README file on your repository.
4. Follow instructions in the README to make your portfolio your own with information about yourself(completeness only) and your own definition of data science (graded for **level 1 process**)
5. complete the `success.md` file as per the instructions in the comments in that file (it is a syllabus quiz)
6. Create a Jupyter notebook called `grading.ipynb` and write a function that computes a grade for this course, with the docstring below. Your notebook will need to follow the [course style guide](#), following style from other courses will not earn credit.
7. In the same notebook, iterate over the example anonymized gradebook below and compute a grade for each one.

[Skip to main content](#)

### Warning

Do not merge your “Feedback” Pull Request

## 1.3.1. Docstring

```
'''
Computes a grade for CSC/DSP310 from numbers of achievements at each level

Parameters:
-----
level1_acheivements : int
    number of level 1 achievements earned
level2_acheivements : int
    number of level 2 achievements earned
level3_acheivements : int
    number of level 3 achievements earned

Returns:
-----
letter_grade : string
    letter grade with possible modifier (+/-)
'''
```

## 1.3.2. Students to check

Treat this list of lists as a gradebook and use your function to compute a grade for each one and store them all to a list.

```
acheivements_only = [[15,14,1],[15,15,10],[12,12,12]]
```

## 1.4. Help


(optional, but useful information)

### 1.4.1. Sample tests

Here are some sample tests you could run to confirm that your function works correctly, they are in one block here, but you should run them one at a time and with text between:

```
assert compute_grade(15,15,15) == 'A'
assert compute_grade(15,15,13) == 'A-'
assert compute_grade(15,14,14) == 'B-'
assert compute_grade(14,14,14) == 'C-'
assert compute_grade(4,3,1) == 'D'
assert compute_grade(15,15,6) == 'B+'
```

[Skip to main content](#)

 W  
rem  
side

## 1.4.2. Notebook Checklist

- a Markdown cell with a heading
- your function called `compute_grade`
- three calls to your function that verify it returns the correct value for different number of badges that produce at three different letter grades.
- markdown cells with explanation of the calls and anything else so that the notebook reads like a report.

## Earning Level 3

Starting in week 3 it is recommended that you spend some time each week working on extensions.

Use the feedback you get on assignments to inspire your extensions.

## 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 versions

### Important

This will be updated; new github features change how it works

## 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, underscores `_` or hyphens `-` are both good. Each `chapter` or file should have a descriptive file name ( `with_no_spaces` ) and descriptive title for it.

## 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!
```
```

```
```{skip}
```

[Skip to main content](#)

```
```
```

```
```{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.

## Generic Extension Ideas

### Extension

If there were parts of your previous assignments that you thought were interesting and you want to work with those data more, you can. You need to do *more complex* analyses of them, but you can build off of what you already have done, especially for assignments 2, 3, and 5.

## Extend Assignment 7, 8, or 9

Assignments 7-9 help you think through what machine learning tasks are. Extend those ideas by adding additional experiments based on your own questions or the questions in your feedback.

## Learn a new model

Repeat what you did in 7, 8, or 9, with a different model.

## Alternatives to Extending Assignments for Level 3

These are other ways you can earn level 3 achievements besides adding onto previous assignments.

If your goal is, for example, a B+ (you need 5 level 3s) you could only do 1-2 skills per portfolio check (there are 4 checks).

## Tutorial

Write a notebook that explains a concept related to a skill with examples in a real dataset and with visuals or a toy dataset (minimal number of columns rows)

## Cheatsheet

Make a detailed reference with code outputs on a topic or a few topics.

## Blog post

Write a blog post styled Notebook that compares or analyzes something, for example:

- how do different ways of loading data compare
- describe best practices you've learned and show why they're good with examples

## Practice Problems and Solutions

Based on the level 3 rubric descriptions, write practice problems that build off of the lecture notes. Include solutions and descriptions for each. These can be open ended or multiple choice questions with plausible distractors. A plausible distractor is an incorrect answer that represents a way that you think someone could misunderstand.

For example if the question is  $37 + 15 = ?$ , MCQ with plausible distractors might be:

- 52 (correct)
- 412 (didn't carry the one, correctly:  $7+5 = 12$ ,  $3+1 = 4$ )

[Skip to main content](#)



Tip

If you  
achieve  
comple  
comple  
work



- 43 (carried one into wrong column,  $7 + 5 = 12$ ,  $1+2 = 3$ ,  $3+1 = 3$ )

## Long single analysis

Collect data from multiple sources, prepare each for analysis, and merge them together then do some exploratory data analysis. Describe each step, interpret all outputs, and put the analysis in context of the Data Science Process.

This would be one long notebook that covers many skills at once.

## Create datasets that fail

Create datasets that violate assumptions of a model we have learned. The [sklearn data generators](#) are a good place to start.

## Build a data set for Prediction

Build a dataset that works for prediction (classification, regression, or clustering) from other sources.

## Organize your knowledge

Develop some sort of visual aid that demonstrates how you understand some aspect(s) of data science working. Think of this as something that future students could use to help them learning, so assume prior knowledge topics covered earlier than the one you are demonstrating.

This could be a concept map, a table that shows how you've traced how something works or any other sort of conceptual tool that helps convey your understanding.

## Try alternative libraries/ tools

One option for workflow level 3 is to use other data science skills and reflect on how what we have learned so far helped you learn a new set of tool as an alternative way to do things.

## Try feature engineering or representation learning

Try different transformations and see how they impact how well a model performs. This could be using `sklearn.feature_extraction` tools or trying different types of neural network layers at the beginning.

## Process Level 3

### Process level 3

Process level 3 is a little different than most of the others. You may be able to work it into an analysis notebook, but likely,

[Skip to main content](#)

---

## Data Science Pipeline Comparisons

Find two different sources that describe the data science pipeline or lifecycle. Write a blog style post that discusses their differences and hypothesizes about why they may be different? Are they for different audiences? Is one domain specific? How do they emphasize different modeling tasks? Include a Recommendation for when you think each one is better

## Write a short story

Write a short story that explains the concepts of data science to demonstrate your understanding of process.

## Media Review

Watch/listen/read to an episode of a high quality<sup>[1]</sup> podcast or other type of media and write a blog style summary and review. Highlight what you learned and how it relates to topics covered in class.

Approved Media:

- [Pod of Asclepius, Fall Series: The Philosophy of Data Science](#)
- Chapter 1 & 2 of [Think like a Data Scientist](#) in particular, if you think these would be helpful to assign as reading or teach from at the beginning of the semester next year.
- Algorithms of Oppression (book)
- Weapons of Math Destruction (book)
- [Coded Bias](#) (film, available on netflix & PBS)

---

<sup>[1]</sup> approved Dr. Brown by creating a pull request to add it to the list on this page that is successfully merged. To create a PR, use the suggest an edit button at the top of this page.

## 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](mailto:brownsarahm)) or TA
- 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?

What time are assignments due?

Can I submit this assignment late if ...?

I don't understand my grade on this assignment

## Git and GitHub

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

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

My command line says I cannot use a password

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

My portfolio won't compile

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













# Code Errors

## Key Error

<bound method

## 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.

### conditional

a logical control to do something, conditioned on something else, for example the `if`, `elif` `else`

### corpus

(NLP) a set of documents for analysis

### DataFrame

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

### dictionary

## 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

[Skip to main content](#)

---

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 in 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 call 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  |
|------------------------|--|
| [val]                  | indexing item val from an object; val is int for iterables, or any for mapping |
| [val : val2]           | slicing elemtns val to val2-1 from a listlike object                           |
| [ item1,item2 ]        | creating a list consisting of item1 and item2                                  |
| (param)                | function calls   |
| (item1,item2)          | defining a tuple of item1 and item2  |
| {item1,item2}          | defining a set of item1 and item2  |
| {key:val1, key2: val2} | 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
```

[Skip to main content](#)

|   | 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.

### Cleaning Examples

- [Messy Artists](#) .csv file, that needs to be cleaned, containing data on artists
- [Messy Wheels](#) .csv file, that needs to be cleaned, containing data on various wheel attractions around the globe
- [Clean Artists](#) .csv file, already cleaned, containing data on artists
- [Clean Wheels](#), .csv file, already cleaned, containing data on various wheel attractions around the globe
- [Women's Rugby](#)
- [Web page metrics](#)
- [data cleaning with open refine on survey data](#) this is a tutorial for cleaning data with another tool, but it demonstrates common problems with data well.
- [data cleaning for ecology](#) this is a tutorial for cleaning data with another tool, but it demonstrates common problems with data well.
- [us solar data](#)
- [NYT Data Preparation document](#)
- [Corporate Reputation Rankings](#)

### 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.

[Skip to main content](#)



- [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](#)
- [character psychometrics](#)
- [weather forecast accuracy](#)

## 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

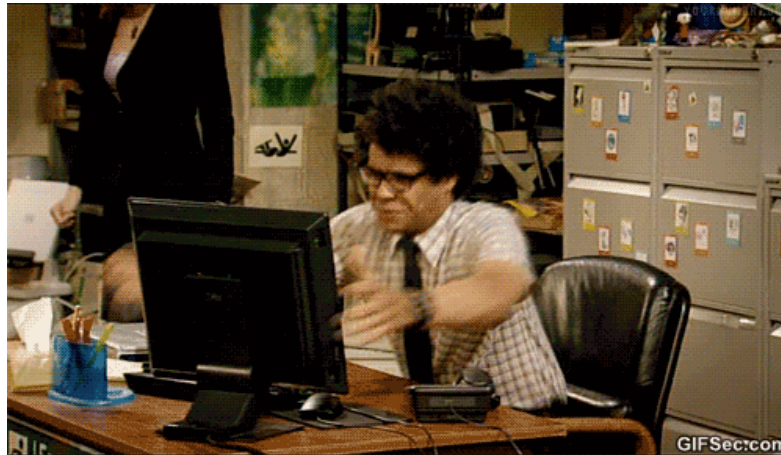
This section is for materials that are not specific to this course, but are likely useful. They are not generally required readings or installs, but are options or advice I provide frequently.

## on email

# How to 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  
programm  
Program  
available  
that linke

### 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

[Skip to main content](#)

hile 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

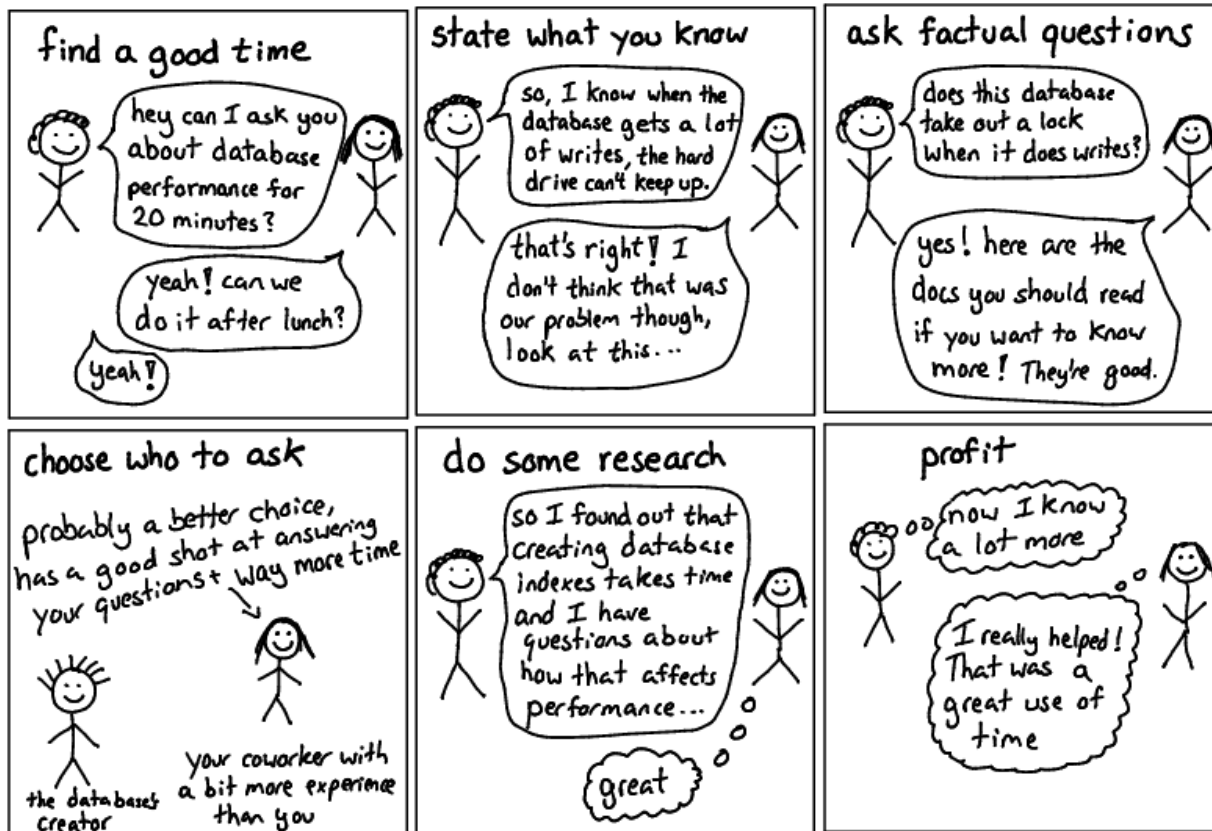
In portfolios, your goal is to extend and apply the concepts taught in class and practiced in assignments to solve more realistic problems. You may also reflect on your learning in order to demonstrate deep understanding. These will require significant reading beyond what we cover in class.

# 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.

## Understanding Errors

Error messages from the compiler are not always straight forward

[Skip to main content](#)

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 fo the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your indepenence 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 realisitc 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 you 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 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 will 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

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[<sup>^</sup>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.