About this Book

Contents

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

This class meets MWF 2-2:50pm in Ranger 302.

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

Course Calendar on BrightSpace

> **Tip**
>
> subscribe to that calendar in your favorite calendar application

Navigating the Sections

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

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

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

Reading each page

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

> **Try it Yourself**
>
> Notes will have exercises marked like this

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

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

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

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

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

About

About this course

Data science exists at the intersection of computer science, statistics, and machine learning. 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.

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 substantial 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 this semester

This semester is a lot of new things for all of us. This course will be completely online all semester, so we will get to use a single instructional format all semester, including when all campus activities move remote after Thanksgiving. I recognize that those last two weeks of the semester may change your obligations with siblings, parents, work, etc. In light of that, we will cover all of the most important topics and you will have the opportunity to achieve all of the course learning outcomes before Thanksgiving. The material in the last two weeks of the semester will be more advanced, likely interesting and definitely useful material, but if your ability to participate in class is less at that time, it will not hurt your grade.

About this syllabus

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

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

About your instructor

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

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

The best way to contact me 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.

> **Note**
>
> Whether you use CSC or DSP does not matter.

Tools and Resources

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

All of the tools and resources below are either:

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

BrightSpace

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

This is also where your grades will appear and how I will post announcements.

For announcements, you can customize how you receive them.

Important

TL;DR [1]

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

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.

Note

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

Course Website

The course manual will have content including the class policies, scheduling, class notes, assignment information, and additional resources. This will be linked from Brightspace and available publicly online at rhodyprog4ds.github.io/BrownFall22/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub

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

Programming Environment

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

Requirements:

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

Warning

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

Note

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

Recommendation:

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

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

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

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

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

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

Optional:

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

Video install instructions for Anaconda:

- Windows

- Mac

On Mac, to install python via environment, this article may be helpful

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

Textbook

The texts for this class are references and for context and will not be a source of assignments. Both are available free online, but are also relatively affordable if you want a hard copy.

Think Like a Data Scientist

It will be a helpful reference and you may be directed there for answers to questions or alternate explanations ot topics.

Python for Data Science is available free online:

Zoom (backup only, Fall 2021 is in person)

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

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

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

For help, you can access the instructions provided by IT.

---

[1] Too long; didn't read.

A tip from Dr. Brown

I use atom, but I decided to use it by downloading both Atom and Sublime and trying different things in each for a week. I liked Atom better after that and I've stuck with it since. I used Atom to write all of the content in this syllabus. VScode will also work, if needed

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 mutiple 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 MWF 2-2:50pm in Ranger 302. 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 page. Portfolio deadlines will be announced at least 2 weeks in advance.

Note

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

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

keyword	skill	Level 1	Level 2	Level 3
python	pythonic code writing	python code that mostly runs, occasional pep8 adherance	python code that reliably runs, frequent pep8 adherance	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 constrast 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 statndard statistics of a whole dataset and grouped data	Compute and interpret various summary statistics of subsets of data
visualize	Visualize data	identify plot types, generate basic plots from pandas	generate multiple plot types with complete labeling with pandas and seaborn	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	prepare data for analysis	identify if data is or is not ready for analysis, potential problems with data	apply data reshaping, cleaning, and filtering as directed	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
evaluate	Evaluate model performance	Explain basic performance metrics for different data science tasks	Apply and interpret basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit, apply, and interpret preselected classification model to a dataset	fit and apply classification models and select appropriate classification models for different contexts
regression	Apply Regression	identify what data that can be used for regression looks like	fit and interpret linear regression models	fit and explain regrularized 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 optimized	Optimize basic model parameters such as model order	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning
compare	compare models	Qualitatively compare model classes	Compare model classes in specific terms and fit models in terms of traditional model performance metrics	Evaluate tradeoffs between different model comparison types
representation	Choose representations and transform data	Identify options for representing text and categorical data in many contexts	Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred fully specified problems with a single tool pipeline	Solve well-strucutred, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

Assignments and Skills

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

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

Warning

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

Portfolios and Skills

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

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

Detailed Checklists

System Message: WARNING/2 (/home/runner/work/BrownFall22/BrownFall22/index.md, line 1)

Non-consecutive header level increase; 0 to 4 [myst.header]

Grading

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

Principles of Grading

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

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

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

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

How it works

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

These map onto letter grades roughly as follows:

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

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

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

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

For each skill these are defined in the Achievement Definition Table

Participation

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

Office Hours

If you miss questions during class, you can make up level 1 achievements in office hours in the following two weeks. You can earn up to 2 level 1 achievements in a single visit to office hours. To earn them in office hours, you will be asked similar questions, but have the opportunity to answer verbally.

Assignments

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

There will be an assignment each week. Assignments will be a chance to analyze data using the new skills and they will build so that you can get continuous practice with the skills. After your assignment is reviewed, you will get qualitative feedback on your work, and an assessment of your demonstration of the targeted skills. In an assignment, you can earn level 1 achievements in the designated skills by including an outline, you can earn level 2 by completing the analysis as prescribed.

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

Assignments are also an opportunity to get a head start on your portfolio. You can propose extensions of the analysis and get feedback on that proposal before you implement it. This way you can have more guidance on what goes in your portfolio and do the work for it continually.

Feedback on assignments is designed to be a two way *conversation* about your work to help you become a better data science programmer. Reading and acknowledging (reply or emoji reaction) your feedback is required. There is a limit of 2 assignments worth of unacknowledged feedback. If you have 2 unacknowledged assignments, your future assignments will not get feedback, they will be considered unsubmitted.

Portfolio Checks

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

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

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

TLDR

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

Detailed mechanics

The table below shows the minimum number of skills at each level to earn each letter grade.

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

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

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

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

The letter grade can be computed as follows

Late work

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

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

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

In this issue, include:

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

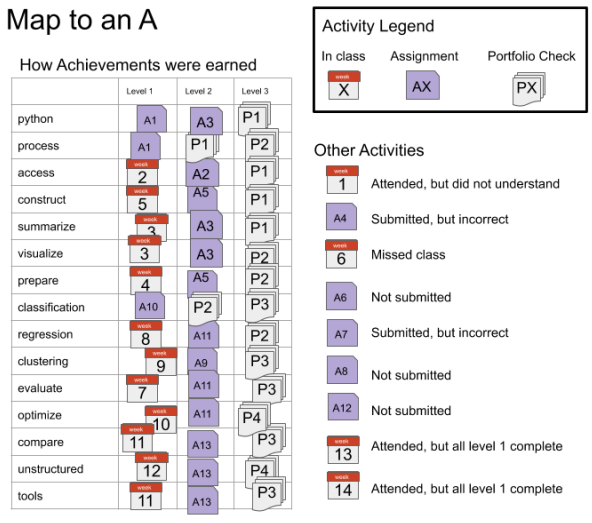
This request should be no more than 7 sentences.

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

Grading Examples

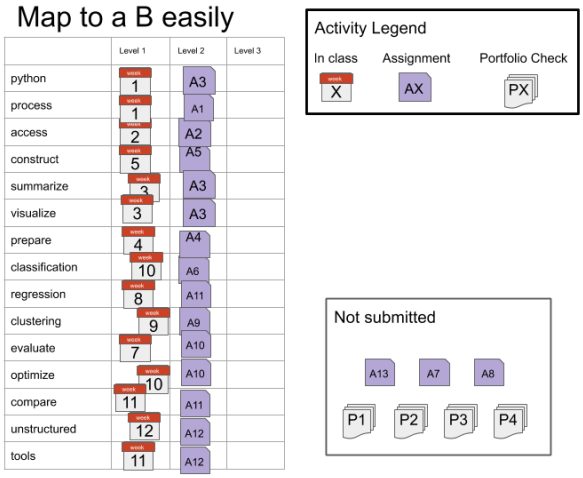
If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check or assignment 13.

Getting an A Without Perfection



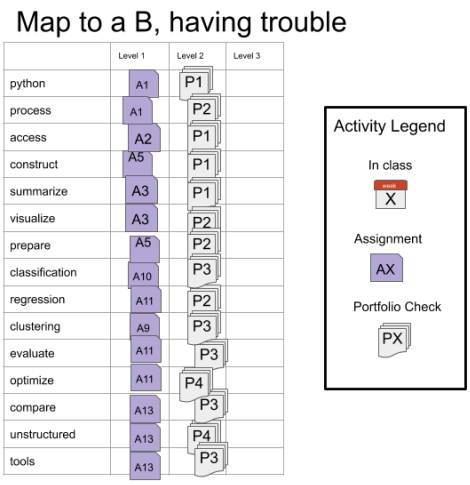
In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level 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



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



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

Grading Policies

Late Work

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

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

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

In this issue, include:

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

This request should be no more than 7 sentences.

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

Regrading

Re-request a review on your Feedback Pull request.

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

For specific questions, reply to a specifc comment.

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

Collaboration

You may talk to other students about the general approach or ask clarifying questions about instructions by posting to the GitHub discussions for our course.

You may only view one another's code, when explicitly instructed to share for peer review, and only *via GitHub* by adding a classmate as a collaborator. If you do not have permission to share your repository or an assignment is not created as a team assignment, then you may not collaborate on that assignment at the code level.

Support

Academic Enhancement Center

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

- **STEM Tutoring** helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses. The STEM Tutoring program offers free online and limited in-person peer-tutoring this fall. Undergraduates in introductory STEM courses have a variety of small group times to choose from and can select occasional or weekly appointments. Appointments and locations will be visible in the TutorTrac system on September 14th, 2020. The TutorTrac application is available through URI Microsoft 365 single sign-on and by visiting aec.uri.edu. More detailed information and instructions can be found on the AEC tutoring page.
- **Academic Skills Development** resources helps students plan work, manage time, and study more effectively. In Fall 2020, all Academic Skills and Strategies programming are offered both online and in-person. UCS160: Success in Higher Education is a one-credit course on developing a more effective approach to studying. Academic Consultations are 30-minute, 1 to 1 appointments that students can schedule on Starfish with Dr. David Hayes to address individual academic issues. Study Your Way to Success is a self-guided web portal connecting students to tips and strategies on studying and time management related topics. For more information on these programs, visit the Academic Skills Page or contact Dr. Hayes directly at davidhayes@uri.edu.
- The **Undergraduate Writing Center** provides free writing support to students in any class, at any stage of the writing process: from understanding an assignment and brainstorming ideas, to developing, organizing, and revising a draft. Fall 2020 services are offered through two online options: 1) real-time synchronous appointments with a peer consultant (25- and 50-minute slots, available Sunday - Friday), and 2) written asynchronous consultations with a 24-hour turn-around response time (available Monday - Friday). Synchronous appointments are video-based, with audio, chat, document-sharing, and live captioning capabilities, to meet a range of accessibility needs. View the synchronous and asynchronous schedules and book online, visit uri.mywconline.com.

General URI Policies

COVID/Viral Illness Precautions Statement

The University is committed to delivering its educational mission while protecting the health and safety of our community. As members of the URI community, students are required to comply with standards of conduct and take precautions to keep themselves and others safe.

> **Important**
>
> Masks are required in all classrooms, laboratories, and spaces where direct academic instruction and research are taking place, unless the instructor or staff member expressly waives that requirement.

> **Warning**
>
> not waived

We strongly recommend surgical or higher grade masks where face coverings are required. Masks should be properly worn, well-fitting, and high quality.

Students who do not comply with the classroom/lab masking requirement will be asked to leave class and will be reported through the Student Conduct process.

Students who are experiencing symptoms of viral illness should NOT go to class/work. Those who test positive for COVID-19 should follow the isolation guidelines from the Rhode Island Department of Health and CDC.

Anti-Bias Statement:

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

Disability Services for Students Statement:

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

Academic Honesty

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

- Using material, directly or paraphrasing, from published sources (print or electronic) without appropriate citation
- Claiming disproportionate credit for work not done independently
- Unauthorized possession or access to exams
- Unauthorized communication during exams
- Unauthorized use of another's work or preparing work for another student
- Taking an exam for another student

- Altering or attempting to alter grades
- The use of notes or electronic devices to gain an unauthorized advantage during exams
- Fabricating or falsifying facts, data or references
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

URI COVID-19 Statement

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

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

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

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

Help Hours and Course Communications

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

Help Hours

Day	Time	Location	Host
Tuesday	11am-12pm	139 Tyler Hall	Aiden
Wednesday	11am-12pm	139 Tyler Hall	Aiden
Wednesday	7pm-8:30pm	Zoom	Dr. Brown
Thursday	11am-12pm	139 Tyler Hall	Aiden
Friday	11am-12pm	139 Tyler Hall	Aiden
Friday	3:30-4:30	134 Tyler Hall	Dr. Brown

To reach out, By usage

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

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

By Platform

Use e-mail for

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

Use github for

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

Use prismia for

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

Tips

For assignment help

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

Using issues

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

For E-email

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

> **Note**
>
> Whether you use CSC or DSP does not matter.

1. Welcome to Programming to 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. How this class will work

Participatory Live Coding

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

Debugging is both technical and a soft skill

1.3. Programming for Data Science vs other Programming

The audience is different, so the form is different.

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

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

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

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

1.4. Get Organized!

In this class you will have many separate folders that your work is in. The separate folders are *required* because we will use GitHub for submission.

I recommend you make a folder for this class and make all of your other folders inside that.

Create a separate notes folder in there too. We will be writing code each class, that you should keep your own notes.

Important

If you made your notebook in a location other than where you want it to be, you can move it like any other file using Finder on mac or File Explorer on Windows.

1.5. Jupyter Notebooks

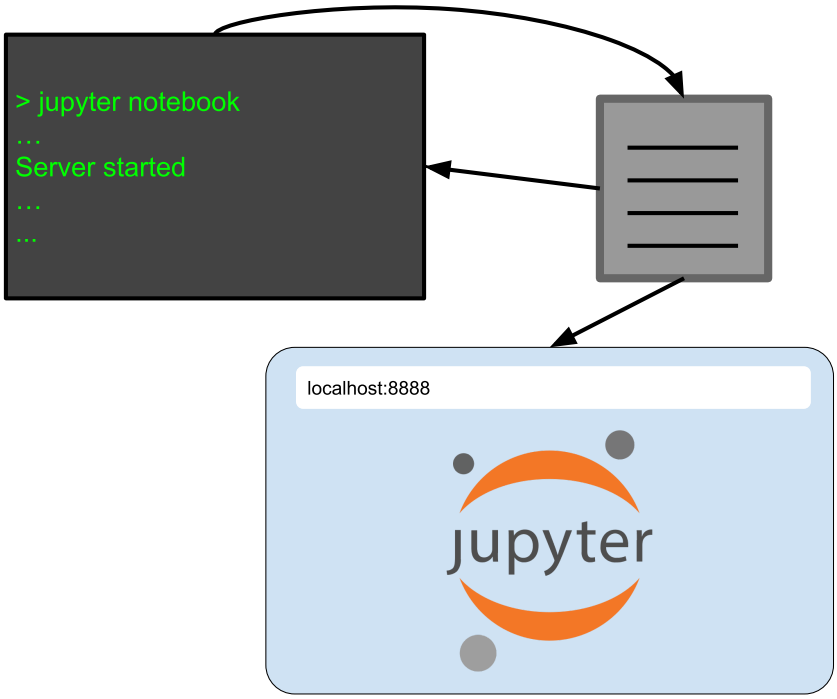
Launch a `jupyter notebook` server:

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

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

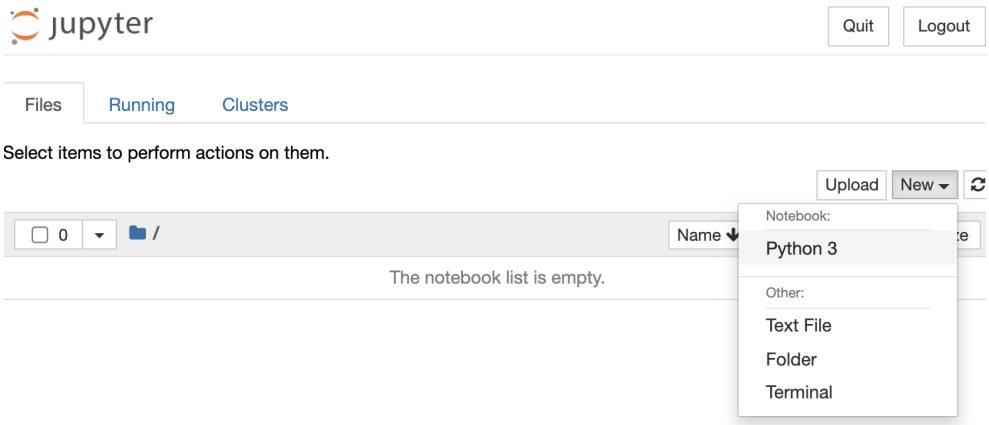
1.5.1. What just happened?

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

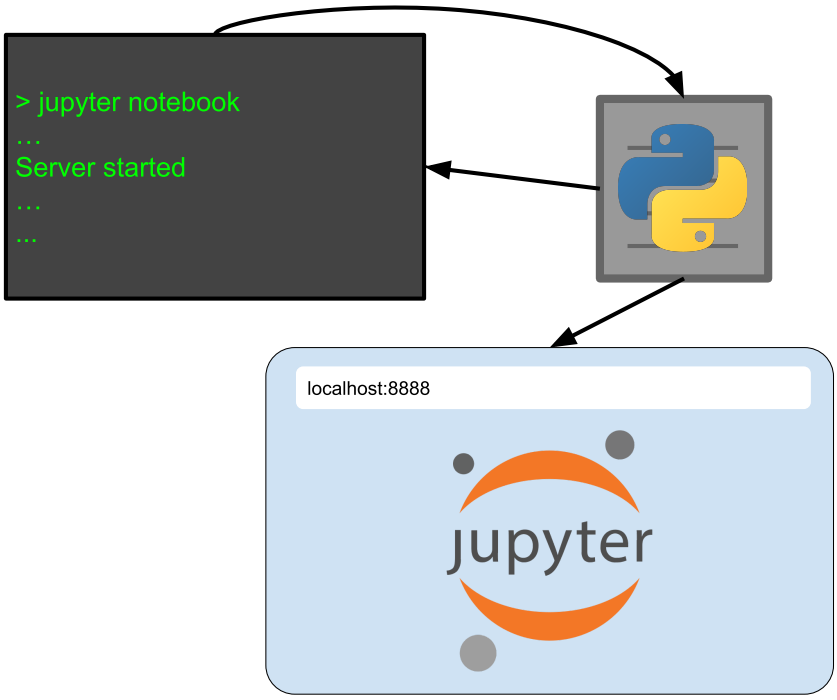


1.5.2. Start a Notebook

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



Now, it starts a python kernel on the webserver

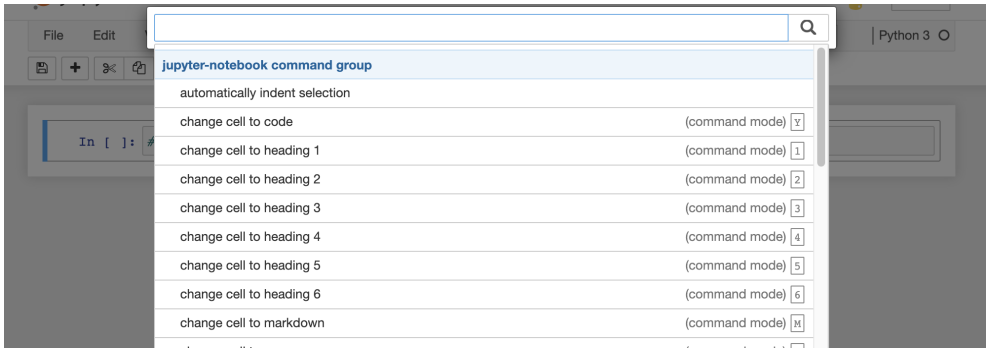


1.5.3. A jupyter notebook tour

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

```
In [ ]: # This is a comment in a code cell
```

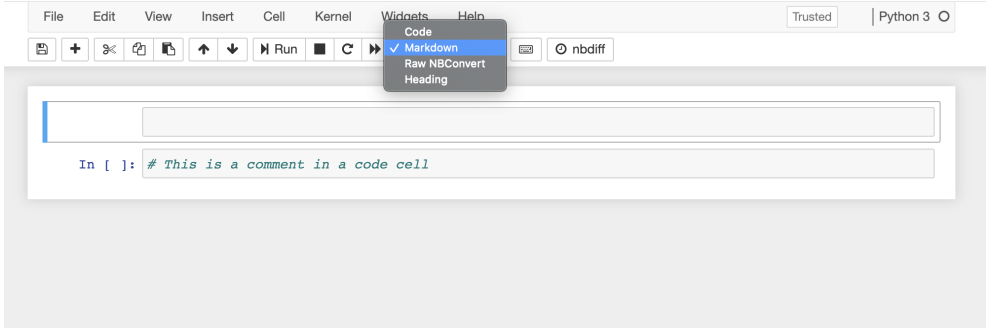
When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.



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

```
In [ ]: # This is a comment in a code cell
```

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

1.5.4. Notebook Reminders

Blue border is command mode, green border is edit mode

use Escape to get to command mode

Common command mode actions:

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

use enter/return to get to edit mode

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

```
4+6
```

```
10
```

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

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

```
'sarahsarahsarah'
```

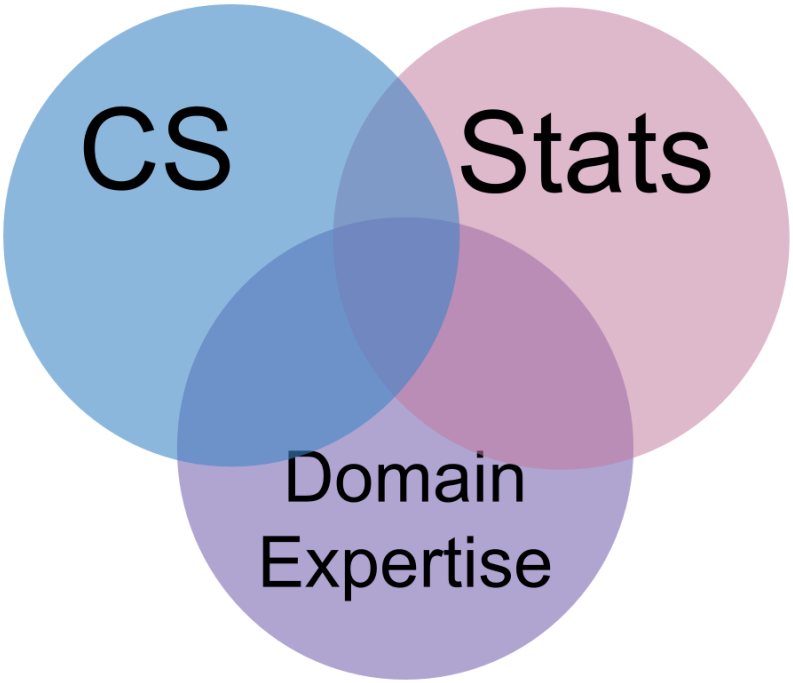
1.6. Getting Help in Jupyter

When your cursor is inside the `()` of a function if you hodl the shift key and press tab it will open a popup with information.

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

```
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.
```



1.7. Course Administration

We will use GitHub for all course administration.

- join to discuss
- go to discussions and introduce yourself
- course website
- Main page with links

1.8. Prepare for the next class

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

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

Think like a data scientist is written for practitioners; not as a text book for a class. It does not have a lot of prerequisite background, but the sections of it that I assign will help you build a better mental picture of what doing Data Science about. In chapter 1, focus most on sections 1.1, 1.3, and 1.7.

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.

1.9. Questions after class

1.9.1. Grading

1.9.1.1. How do the portfolios work? what are the 1/2/3 achievement levels? How is gradint structured?

Read the syllabus carefully and we will discuss on Friday

1.9.1.2. Will there be any group work?

Not in the regular sense of collaborative and shared grade. There will be optional collaboration opportunities and in class discussion/troubleshooting together.

1.9.2. Uses for what we cover

1.9.2.1. How in depth does Data Science go, and what can it be used for in the industry?

The basic ideas here can be used for any tabular data in industry exactly as we will cover them. At the end of the semester, we will see in less detail how to work with text and images, with a focus on translating what you learned on tabular data (because its low dimensional and easier to see/faster to process) to more complex data.

We'll talk more about this on Friday and every time we use a new dataset in class.

1.9.2.2. What is the difference between data analytics and data science? Are analysts and scientists jobs different?

This is a hard question. It varies company to company. It is, however, a topic a lot of Data Science Bloggers write about. If you find some you like, share them on the discussion board or submit a PR.

1.9.3. Logistics

1.9.3.1. What is the main site

this is it

1.9.3.2. do these prismia chats stay up after class or go away?

They persist. You can scroll back or get a transcript by clicking the > in the top left, then the 3 bar menu icon and then “Get Transcript From Class”

1.9.4. Tools

1.9.4.1. Can we open jupyter notebook server without the terminal?

Yes, you can, but I do not use it that way and they change how that works from time to time, so I can only troubleshoot with you via the terminal.

1.9.4.2. Which IDE to use? Can we use VSCode?

Your will be required to submit jupyter notebooks that are compatible with some other jupyter related tools I use to process them for grading.

1.9.4.3. Will we all have a single server to have various notebooks for assignments, or will we be required to make different servers for other assignments?

The jupyter notebook web server is something that you will start and stop many times, each working session you'll stop it. You can run multiple in parallel or use one and open multiple notebooks.

1.9.4.4. Will we do any web scraping in this class?

Yes

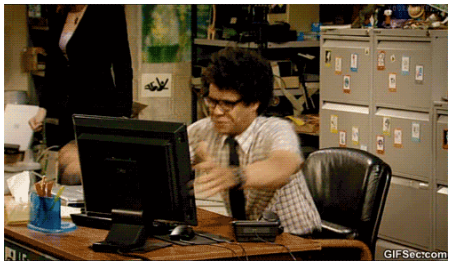
1.9.4.5. Will we be able to choose our own data sets when doing assignments?

Yes, with some requirements, on most assignments.

2. Data Science and Course Logistics

2.1. Course Logistics

Class is designed to avoid this:



Read more about how I'm designing this course to help you learn on the how to learn page.

2.2. Check your understaning of the syllabus

It's easy when reading something long to lose track of it. Your eyes can go over each word, without actually retaining the information, but it's important to understand the syllabus for the course.

You can find the answers to the following questions on the syllabus. If you've already read it, try answering them to check your understanding. If you haven't read it yet, use these to guide you to get familiar with finding key facts about the course on the syllabus.

1. What do you need to bring to class each day?
2. What is the basis of grading for this course?
3. How do you reference the course text?
4. What is the penalty for missing an assignment?

More information about the course is available throughout the site, the next few questions will help you self-check that you've found the important things. Remember, the goal is not necessarily to memorize all of this, but to be able to find it.

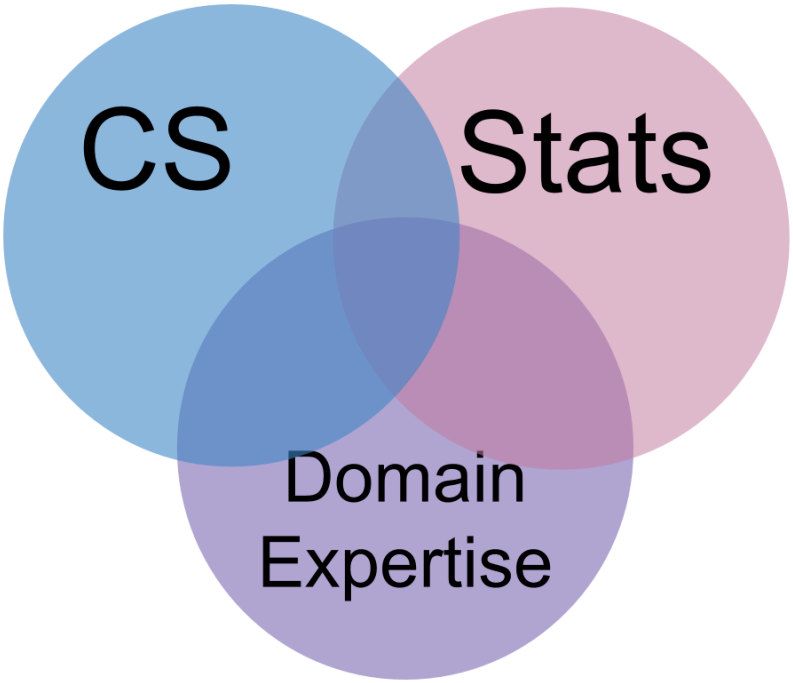
1. When & what are you expected to read for this class?

- [ ] read the text book before class
- [ ] review notes & documentation after class
- [ ] preview the notes & documentation before class
- [ ] read documentation and text book after class

1. Your assignment says to find a dataset that has variables of a specific type, which website can you use?
2. Your assignment says to find a dataset of any type about something you're interested in, which resource would you use?

2.3. What is Data Science?

Data Science is the combination of

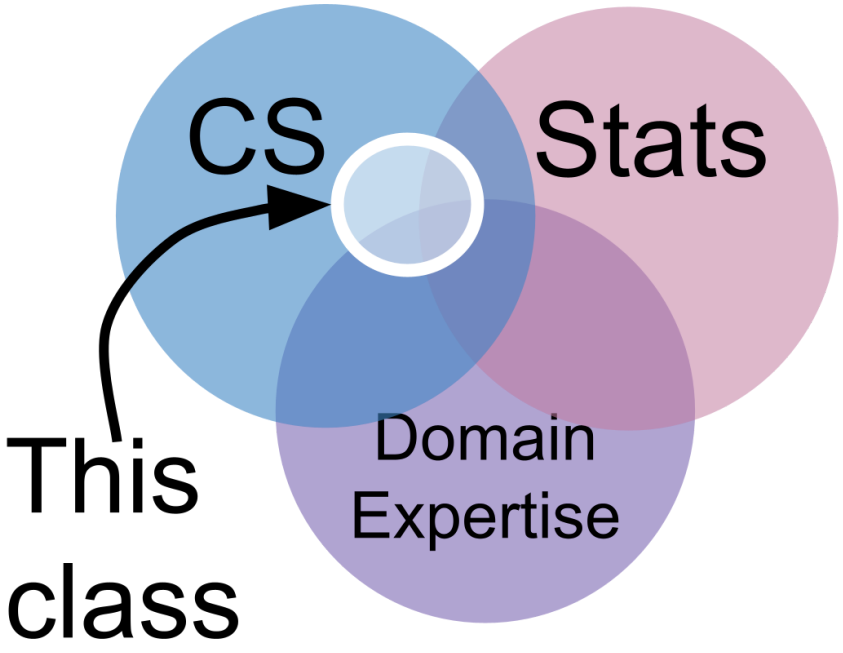


statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

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.

2.3.1. In this class,

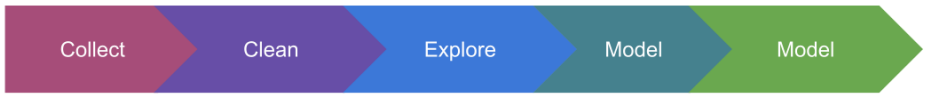


We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

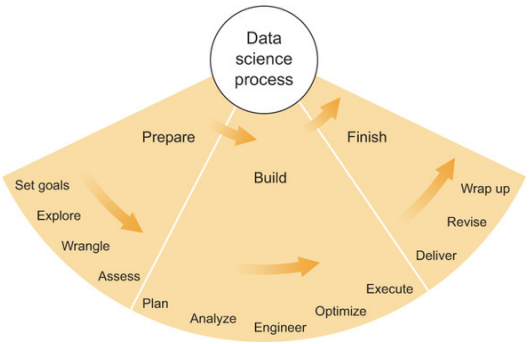
But there are many definitions. We'll use this one, but you may come across others.

2.3.2. How does data science happen?

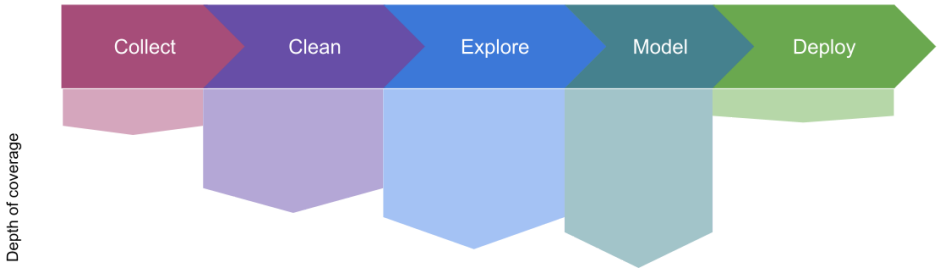
The most common way to think about what doing data science means is to think of this pipeline. It is in the perspective of the data, these are all of the things that happen to the data.



Another way to think about it

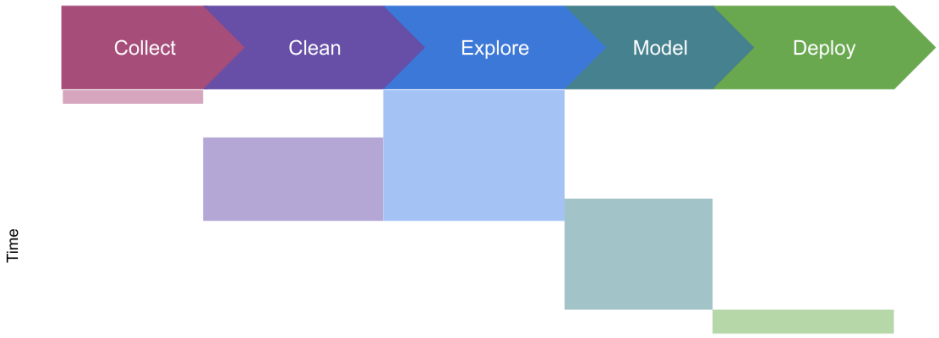


2.3.3. how we'll cover Data Science, in depth



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

2.3.4. 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.

3. Reading Docstrings, Object Inspection, and Loading Data

3.1. Programming is a Practice

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

Given this code excerpt, how could you print out "Sarah_Brown"?

```
first = 'Sarah'
last = 'Brown'
```

We can print the docstring out, as a whole instead of using the shfit + tab to view it.

```
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 `value, ..., sep`

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.

```
print(first,last)
```

```
Sarah Brown
```

```
print(first,last,sep='_')
```

```
Sarah_Brown
```

```
type(first)
```

```
str
```

```
def compute_grade(num_level1,num_level2,num_level3):
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    -----------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with modifier (+/-)
    '''
    if num_level1 == 15:
        if num_level2 == 15:
            if num_level3 == 15:
                grade = 'A'
            elif num_level3 >= 10:
                grade = 'A-'
            elif num_level3 >=5:
                grade = 'B+'
            else:
                grade = 'B'
        elif num_level2 >=10:
            grade = 'B-'
        elif num_level2 >=5:
            grade = 'C+'
        else:
            grade = 'C'
    elif num_level1 >= 10:
        grade = 'C-'
    elif num_level1 >= 5:
        grade = 'D+'
    elif num_level1 >=3:
        grade = 'D'
    else:
        grade = 'F'


    return grade
```

```
type(compute_grade)
```

```
function
```

```
help(compute_grade())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1845/3187590259.py in <module>
----> 1 help(compute_grade())

TypeError: compute_grade() missing 3 required positional arguments: 'num_level1',
'num_level2', and 'num_level3'
```

3.1.1. Why inspection in code?

Some IDEs give you GUI based tools to inspect objects. We are going to do it programmatically inline with our analyses for two reasons.

(minor, logistical) it helps make for good notes (most importantly) it helps build habits of data science

3.1.2. Investigating how doc strings work

We can see how the docstring impacts help and how exactly it has to be formatted to become a docstring

```
def ex_1(a):
    print(a)
```

```
help(ex_1())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1845/1576687116.py in <module>
----> 1 help(ex_1())

TypeError: ex_1() missing 1 required positional argument: 'a'
```

```
def ex_2(a):
    #nis htis a docstring?
    print(a)
```

```
help(ex_2())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1845/3614324986.py in <module>
----> 1 help(ex_2())

TypeError: ex_2() missing 1 required positional argument: 'a'
```

```
def ex_3(a):
    ''' this is a docstring'''
    #nis htis a docstring?
    print(a)
```

```
help(ex_3())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1845/980374236.py in <module>
----> 1 help(ex_3())

TypeError: ex_3() missing 1 required positional argument: 'a'
```

```
def ex_4(a):
    """this is a docstring"""
    #nis htis a docstring?
    print(a)
```

```
help(ex_4())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1845/2845574773.py in <module>
----> 1 help(ex_4())

TypeError: ex_4() missing 1 required positional argument: 'a'
```

Tip

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

3.2. Coffee Data

Structured data is easier to work with than other data.

We're going to focus on tabular data for now. At the end of the course, we'll examine images, which are structured, but more complex and text, which is much less structured.

We're going to use a dataset about coffee quality today.

How was this dataset collected?

- reviews added to DB
- then scraped

Where did it come from?

- coffee Quality Institute's trained reviewers.

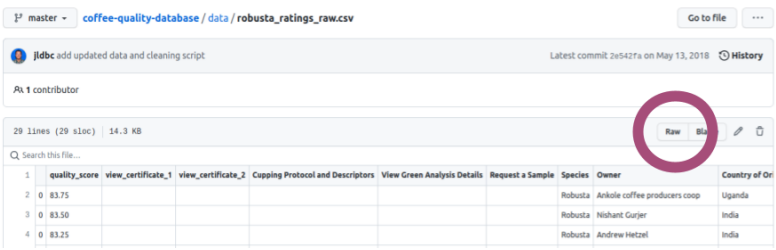
what format is it provided in?

- csv (Comma Separated Values)

what other information is in this repository?

- the code to scrape and clean the data
- the data before cleaning

Get raw url for the dataset click on the raw button on the csv page, then copy the url.



Important

It's important to always know where data came from and how it was collected.

This helps you know what is is useful for and what its limitations are.

An important research article on documenting datasets for machine learning is called Datasheets for Datasets these researchers also did a follow up study to better understand how practitioner use datasheets and decide how to use data.

Note

If this type of research is interesting to you, let me know!

We'll save that url as a variable to work with it.

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
```

3.3. Loading in Data

We will use a library called Pandas

```
import pandas as pd
```

- the `import` keyword is used for loading packages
- `pandas` is the name of the package that is installed
- `as` keyword allows us to assign an alias (nickname)
- `pd` is the typical alias for pandas

> **Further Reading**
>
> the pandas package overview can be a good read if reading more about things helps you understand

We can read data in using the read csv file

```
pd.read_csv(coffee_data_url)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Ex
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	A
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	A
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	M
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	25
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	la esperanza, municipio juchique de ferrer, ve...	1104328663	terra mia	900	...	None	20	Se 17
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	coeb koperativ ekselsyo basen (350 members)	NaN	haiti coffee	~350m	...	Blue-Green	16	M
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	beneficio atlantic condega	017-053-0211/ 017-053-0212	exportadora atlantic s.a	1100	...	Green	5	J
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11/853/165	unicafe	4650	...	Green	4	M
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigrah s.a de c.v.	13-111-053	cigrah s.a de c.v	1400	...	Green	2	Aj

1311 rows × 44 columns

This read in the data and prints it out because it is the last line on the cell. If we do something else after, it will read it in, but not print it out

```
pd.read_csv(coffee_data_url)

print(first)
```

```
Sarah
```

In order to use it, we save the output to a variable.

```
coffee_data = pd.read_csv(coffee_data_url)
```

Then we can check the type.

```
type(coffee_data)
```

```
pandas.core.frame.DataFrame
```

This is a new type that is provided by the pandas library. Notice this uses the full libary name, not the alias, because this comes from the code for the library itself, not our current code where pandas as a nickname.

```
coffee_df = pd.read_csv(coffee_data_url)
```

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Ex
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	A
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	A
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	M
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	25
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	la esperanza, municipio juchique de ferrer, ve...	1104328663	terra mia	900	...	None	20	Se 17
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	coeb koperativ ekselsyo basen (350 members)	NaN	haiti coffee	~350m	...	Blue-Green	16	M
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	beneficio atlantic condega	017-053-0211/ 017-053-0212	exportadora atlantic s.a	1100	...	Green	5	J
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11/853/165	unicafe	4650	...	Green	4	M
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigrah s.a de c.v.	13-111-053	cigrah s.a de c.v	1400	...	Green	2	A

1311 rows × 44 columns

If you're curious about something, try it out, see what happens. We're going to use a lot of code inspection tools during class. These are helpful both for understanding what's going on, but the advantage to knowing how to get this information programmatically even though a different IDE would give you inspection tools is that it helps you treat your code as data.

3.4. Good Code is always relative

> **Important**
>
> I added this section as notes, that was not in class today. I said similar things last week, but this includes more references and context.

In programming for data science, we are often trying to tell a story.

> **Try it yourself**
>
> How might this goal change your code for this class relative to other code you have written or could imagine writing?

Python is a fully open source project and as such is governed by community standards and conventions.

> **Try it yourself**
>
> Find PEP8 (note that following it is part of earning python achievements)

The documentation for the full language is online too.

Guido van Rossum was the first main developer and wrote essays about python too.

it's pretty popular

3.5. Questions After Class

3.5.1. About the Course

3.5.1.1. Will we further go over how to achieve level 3 achievements with more specificity?

Right now, you are still only able to earn level 1s and then with assignment 2 you can start earning level 2s. After that, it will make morse sense to be able to talk about portfolios.

3.5.1.2. How do portfolio checks work?

At a logistical level, you add files to your portfolio repository by a specific check date and then I grade them.

3.5.1.3. how much stats will we do in this class?

Only a little bit. We wil do some modeling of data and compute basic statistics, but we will not cover the underlying concepts of statistics, much. However if you know some statistics, you will be able to extend what we cover to use them.

3.5.1.4. Is there a rate at which we need to complete skill checks if we fall behind?

Follow the table on the Achievements page for which assignments and portfolio checks have which achievements eligible. Some assignments you can earn only 1-2 achievements others you can earn 4-5 level 2s. It is not recommended that you skip early chances because there are future chances, though. However, sometimes if you have an achievement already you can skip a section of an assignment.

3.5.2. About Jupyter

3.5.2.1. Can you interact with those data tables that we put on jupyter today in real time?

Yes, we can manipulate the data, but we read a copy in. We are not manipulating the version that was on GitHub.

3.5.2.2. Will we eventually learn how to filter data in order to separate different names of data, or perform mathematical operations on the datasets?

Yes, all!

3.5.2.3. Can you show how to launch a book another day after we saved it and come back to it?

When you launch your server on the “home” tab you can click on the file name of a previously saved notebook to work on it again.

3.5.2.4. can you just put anything in the docstring?

Yes, technically, but you should follow good code style guidelines.

3.5.2.5. What would happen if we just call ‘pd.read_csv(coffe_data_url)’ instead of storing it in a variable and then call the variable?

It would print it out, but then you don’t have a variable, so you would have to read it in again to be able to manipulate it.

3.5.2.6. What is considered scraped data?

Data that is pulled from websites automatically. We will do some web scraping starting this week.

3.5.3. Questions we will answer in the rest of this week

- How to view a more detailed look of the data instead of it only showing the first and last few columns
- Could we retrieve data in all formats with one function?
- Can we access data like a 2d array?
- What is the function to check the unique values in a column?

4. Data Frames and other iterables

Today, we’re going to explore DataFrames in greater detail. We’ll continue using that same coffee dataset.

```
import pandas as pd
```

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

```
coffee_df =pd.read_csv(coffee_data_url)
```

Important

A reason to use Jupyter is that it formats the output to be more readable. Compare the view of the DataFrame with jupyter and without.

Jupyter uses the object’s `to_html` method if it exists, where the `print` function casts the object to a string.

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	...	Green	2
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	...	NaN	2
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	...	Green	0
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	...	Green	7
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	...	Green	3
5	6	Robusta	andrew hetzel	India	NaN	NaN	(self)	NaN	cafemakers, llc	3000'	...	Green	0
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Green	0
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/18	kaapi royale	3140	...	Bluish-Green	0
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/17	kaapi royale	1000	...	Green	0
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	0	ugacof ltd	900-1300	...	Green	6
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1095	...	Green	1
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/12	kaapi royale	1000	...	Green	0
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Green	1
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	0	kasozi coffee farmers association	1367	...	Green	7
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	0	ankole coffee producers coop	1488	...	Green	2
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	...	Green	0
16	17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	...	Blue-Green	0
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	0	kawacom uganda ltd	1600	...	Green	1
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	0	nitubaasa ltd	1745	...	Green	2
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	0	mannya coffee project	1200	...	Green	1
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Bluish-Green	1
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	...	Green	0
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	3000'	...	Green	0
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	...	Blue-Green	1
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	...	Blue-Green	0
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	...	NaN	6
26	27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	...	Green	1
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	...	None	9

28 rows × 44 columns

```
print(coffee_df)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	\
0	1	Robusta	ankole coffee producers coop	Uganda	
1	2	Robusta	nishant gurjer	India	
2	3	Robusta	andrew hetzel	India	
3	4	Robusta	ugacof	Uganda	
4	5	Robusta	katuka development trust ltd	Uganda	
5	6	Robusta	andrew hetzel	India	
6	7	Robusta	andrew hetzel	India	
7	8	Robusta	nishant gurjer	India	
8	9	Robusta	nishant gurjer	India	
9	10	Robusta	ugacof	Uganda	
10	11	Robusta	ugacof	Uganda	
11	12	Robusta	nishant gurjer	India	
12	13	Robusta	andrew hetzel	India	
13	14	Robusta	kasozi coffee farmers association	Uganda	

```
14          15  Robusta       ankole coffee producers coop            Uganda
15          16  Robusta                     andrew hetzel             India
16          17  Robusta                     andrew hetzel             India
17          18  Robusta                kawacom uganda ltd            Uganda
18          19  Robusta                     nitubaasa ltd            Uganda
19          20  Robusta              mannya coffee project            Uganda
20          21  Robusta                     andrew hetzel             India
21          22  Robusta                     andrew hetzel             India
22          23  Robusta                     andrew hetzel     United States
23          24  Robusta                       luis robles           Ecuador
24          25  Robusta                       luis robles           Ecuador
25          26  Robusta                       james moore     United States
26          27  Robusta                     cafe politico             India
27          28  Robusta                     cafe politico           Vietnam

                            Farm.Name Lot.Number  \
0       kyangundu cooperative society        NaN
1      sethuraman estate kaapi royale         25
2                 sethuraman estate        NaN
3                ugacof project area        NaN
4   katikamu capca farmers association        NaN
5                                NaN        NaN
6                 sethuraman estates        NaN
7      sethuraman estate kaapi royale          7
8                 sethuraman estate        RKR
9                           ishaka        NaN
10               ugacof project area        NaN
11     sethuraman estate kaapi royale      RC AB
12                sethuraman estates        NaN
13             kasozi coffee farmers        NaN
14            kyangundu coop society        NaN
15                sethuraman estate        NaN
16                sethuraman estates        NaN
17                          bushenyi        NaN
18   kigezi coffee farmers association        NaN
19              mannya coffee project        NaN
20                sethuraman estates        NaN
21                sethuraman estates        NaN
22                sethuraman estates        NaN
23                        robustasa   Lavado 1
24                        robustasa   Lavado 3
25                  fazenda cazengo        NaN
26                               NaN        NaN
27                               NaN        NaN

                                   Mill         ICO.Number  \
0                 ankole coffee producers                0
1                       sethuraman estate    14/1148/2017/21
2                                     NaN               0000
3                                  ugacof                  0
4                katuka development trust                  0
5                                 (self)                NaN
6                                     NaN                NaN
7                       sethuraman estate    14/1148/2017/18
8                       sethuraman estate    14/1148/2016/17
9                            nsubuga umar                  0
10                                 ugacof                  0
11                      sethuraman estate    14/1148/2016/12
12                                    NaN                NaN
13                                    NaN                  0
14  ankole coffee producers coop union ltd                  0
15                                    NaN               0000
16                     sethuraman estates                NaN
17                                kawacom                  0
18                               nitubaasa                  0
19                  mannya coffee project                  0
20                                    NaN                NaN
21                     sethuraman estates                NaN
22                     sethuraman estates                NaN
23                            our own lab                NaN
24                        own laboratory                 NaN
25                          cafe cazengo                 NaN
26                                    NaN  14-1118-2014-0087
27                                    NaN                NaN

                           Company    Altitude  ...         Color  \
0       ankole coffee producers coop        1488  ...         Green
1                       kaapi royale        3170  ...           NaN
2                 sethuraman estate       1000m  ...         Green
3                       ugacof ltd        1212  ...         Green
4       katuka development trust ltd   1200-1300  ...         Green
5                     cafemakers, llc       3000'  ...         Green
6                         cafemakers        750m  ...         Green
7                       kaapi royale        3140  ...  Bluish-Green
8                       kaapi royale        1000  ...         Green
9                         ugacof ltd    900-1300  ...         Green
10                        ugacof ltd        1095  ...         Green
11                      kaapi royale        1000  ...         Green
12                        cafemakers        750m  ...         Green
13  kasozi coffee farmers association        1367  ...         Green
14      ankole coffee producers coop        1488  ...         Green
15                sethuraman estate       1000m  ...         Green
16                    cafemakers, llc        750m  ...    Blue-Green
17                kawacom uganda ltd        1600  ...         Green
18                   nitubaasa ltd        1745  ...         Green
19             mannya coffee project        1200  ...         Green
20                        cafemakers        750m  ...  Bluish-Green
21                    cafemakers, llc        750m  ...         Green
22                    cafemakers, llc       3000'  ...         Green
23                         robustasa         NaN  ...    Blue-Green
24                         robustasa          40  ...    Blue-Green
25           global opportunity fund  795 meters  ...           NaN
26                      cafe politico         NaN  ...         Green
27                      cafe politico         NaN  ...          None

   Category.Two.Defects           Expiration  \
0                     2      June 26th, 2015
1                     2   October 31st, 2018
2                     0     April 29th, 2016
3                     7      July 14th, 2015
4                     3      June 26th, 2015
5                     0  February 28th, 2013
6                     0       May 15th, 2015
7                     0   October 25th, 2018
8                     0    August 17th, 2017
9                     6     August 5th, 2015
10                    1      June 26th, 2015
11                    0    August 23rd, 2017
12                    1       May 19th, 2015
13                    7      July 14th, 2015
14                    2      July 14th, 2015
15                    0     April 29th, 2016
16                    0       June 3rd, 2014
17                    1      June 27th, 2015
18                    2      June 27th, 2015
19                    1      June 27th, 2015
20                    1       May 19th, 2015
21                    0      June 20th, 2014
22                    0  February 28th, 2013
23                    1   January 18th, 2017
24                    0   January 18th, 2017
```

```
25                  6  December 23rd, 2015
26                  1    August 25th, 2015
27                  9    August 25th, 2015

                   Certification.Body  \
0   Uganda Coffee Development Authority
1         Specialty Coffee Association
2         Specialty Coffee Association
3   Uganda Coffee Development Authority
4   Uganda Coffee Development Authority
5         Specialty Coffee Association
6         Specialty Coffee Association
7         Specialty Coffee Association
8         Specialty Coffee Association
9   Uganda Coffee Development Authority
10  Uganda Coffee Development Authority
11        Specialty Coffee Association
12        Specialty Coffee Association
13  Uganda Coffee Development Authority
14  Uganda Coffee Development Authority
15        Specialty Coffee Association
16        Specialty Coffee Association
17  Uganda Coffee Development Authority
18  Uganda Coffee Development Authority
19  Uganda Coffee Development Authority
20        Specialty Coffee Association
21        Specialty Coffee Association
22        Specialty Coffee Association
23        Specialty Coffee Association
24        Specialty Coffee Association
25        Specialty Coffee Association
26        Specialty Coffee Association
27        Specialty Coffee Association

                     Certification.Address  \
0   e36d0270932c3b657e96b7b0278dfd85dc0fe743
1   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
2   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
3   e36d0270932c3b657e96b7b0278dfd85dc0fe743
4   e36d0270932c3b657e96b7b0278dfd85dc0fe743
5   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
6   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
7   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
8   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
9   e36d0270932c3b657e96b7b0278dfd85dc0fe743
10  e36d0270932c3b657e96b7b0278dfd85dc0fe743
11  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
12  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
13  e36d0270932c3b657e96b7b0278dfd85dc0fe743
14  e36d0270932c3b657e96b7b0278dfd85dc0fe743
15  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
16  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
17  e36d0270932c3b657e96b7b0278dfd85dc0fe743
18  e36d0270932c3b657e96b7b0278dfd85dc0fe743
19  e36d0270932c3b657e96b7b0278dfd85dc0fe743
20  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
21  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
22  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
23  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
24  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
25  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
26  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
27  ff7c18ad303d4b603ac3f8cff7e611ffc735e720

                     Certification.Contact unit_of_measurement  \
0   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
1   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
2   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
3   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
4   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
5   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
6   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
7   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
8   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
9   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
10  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
11  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
12  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
13  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
14  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
15  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
16  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
17  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
18  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
19  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
20  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
21  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
22  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
23  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
24  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
25  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
26  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
27  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m

   altitude_low_meters altitude_high_meters altitude_mean_meters
0               1488.0               1488.0                1488.0
1               3170.0               3170.0                3170.0
2               1000.0               1000.0                1000.0
3               1212.0               1212.0                1212.0
4               1200.0               1300.0                1250.0
5               3000.0               3000.0                3000.0
6                750.0                750.0                 750.0
7               3140.0               3140.0                3140.0
8               1000.0               1000.0                1000.0
9                900.0               1300.0                1100.0
10              1095.0               1095.0                1095.0
11              1000.0               1000.0                1000.0
12               750.0                750.0                 750.0
13              1367.0               1367.0                1367.0
14              1488.0               1488.0                1488.0
15              1000.0               1000.0                1000.0
16               750.0                750.0                 750.0
17              1600.0               1600.0                1600.0
18              1745.0               1745.0                1745.0
19              1200.0               1200.0                1200.0
20               750.0                750.0                 750.0
21               750.0                750.0                 750.0
22              3000.0               3000.0                3000.0
23                 NaN                  NaN                   NaN
24                40.0                 40.0                  40.0
25               795.0                795.0                 795.0
26                 NaN                  NaN                   NaN
27                 NaN                  NaN                   NaN

[28 rows x 44 columns]
```

4.1. Examining the Structure of a Data Frame

I told you this was a DataFrame, but we can check with type.

```
type(coffee_df)
```

```
pandas.core.frame.DataFrame
```

We can also see that the DataFrame type comes from the `pandas` library, without the library loaded this type does not exist.

We can also exmaine its parts. It consists of several; first the column headings

```
coffee_df.columns
```

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Fragrance...Aroma', 'Flavor', 'Aftertaste',
       'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup',
       'Clean.Cup', 'Balance', 'Cupper.Points', 'Total.Cup.Points', 'Moisture',
       'Category.One.Defects', 'Quakers', 'Color', 'Category.Two.Defects',
       'Expiration', 'Certification.Body', 'Certification.Address',
       'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
       'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

These are a special type called Index

```
type(coffee_df.columns)
```

```
pandas.core.indexes.base.Index
```

These are still iterable, much like python lists.

and it stores the data

```
coffee_df.values
```

```
array([[1, 'Robusta', 'ankole coffee producers coop', ..., 1488.0,
        1488.0, 1488.0],
       [2, 'Robusta', 'nishant gurjer', ..., 3170.0, 3170.0, 3170.0],
       [3, 'Robusta', 'andrew hetzel', ..., 1000.0, 1000.0, 1000.0],
       ...,
       [26, 'Robusta', 'james moore', ..., 795.0, 795.0, 795.0],
       [27, 'Robusta', 'cafe politico', ..., nan, nan, nan],
       [28, 'Robusta', 'cafe politico', ..., nan, nan, nan]], dtype=object)
```

It also has an index (first column, visually) but it is special because this is how you can index the data.

```
coffee_df.index
```

```
RangeIndex(start=0, stop=28, step=1)
```

Right now this is an autogenerated index, but we can also use the `index_col` parameter to set that up front.

```
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
coffee_df
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	central	...	Green	7
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	luwero central region	...	Green	3
6	Robusta	andrew hetzel	India	NaN	NaN	(self)	NaN	cafemakers, llc	3000'	chikmagalur	...	Green	0
7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Green	0
8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/18	kaapi royale	3140	chikmagalur karnataka india	...	Bluish-Green	0
9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/17	kaapi royale	1000	chikmagalur karnataka	...	Green	0
10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	0	ugacof ltd	900-1300	western	...	Green	6
11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1095	iganga namadrope eastern	...	Green	1
12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/12	kaapi royale	1000	chikmagalur karnataka	...	Green	0
13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Green	1
14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	0	kasozi coffee farmers association	1367	eastern	...	Green	7
15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	0	ankole coffee producers coop	1488	south western	...	Green	2
16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0
17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	chikmagalur	...	Blue-Green	0
18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	0	kawacom uganda ltd	1600	western	...	Green	1
19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	0	nitubaasa ltd	1745	western	...	Green	2
20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	0	mannya coffee project	1200	southern	...	Green	1
21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Bluish-Green	1
22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	chikmagalur	...	Green	0
23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	3000'	chikmagalur	...	Green	0
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	san juan, playas	...	Blue-Green	1
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	san juan, playas	...	Blue-Green	0
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	kwanza norte province, angola	...	NaN	6
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	NaN	...	Green	1
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	NaN	...	None	9

28 rows × 43 columns

```
coffee_df.index
```

```
Int64Index([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
            18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28],
           dtype='int64')
```

Now it's neater

4.2. Extracting Parts of Data Frames

We can look at the first 5 rows with `head`

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	E
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2	
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	central	...	Green	7	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	luwero central region	...	Green	3	

5 rows × 43 columns

and the last 5 with `tail`

```
coffee_df.tail()
```

> **Try it yourself**
>
> How can you look at the first 3 or last 2 rows?

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expiration	Cer
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	san juan, playas	...	Blue-Green	1	January 18th, 2017	
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	san juan, playas	...	Blue-Green	0	January 18th, 2017	
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	kwanza norte province, angola	...	NaN	6	December 23rd, 2015	
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	NaN	...	Green	1	August 25th, 2015	
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	NaN	...	None	9	August 25th, 2015	

5 rows × 43 columns

the shape of a DataFrame is an attribute

```
coffee_df.shape
```

```
(28, 43)
```

```
len(coffee_df)
```

```
28
```

We can pick out columns by name.

```
coffee_df['Species']
```

```
1     Robusta
2     Robusta
3     Robusta
4     Robusta
5     Robusta
6     Robusta
7     Robusta
8     Robusta
9     Robusta
10    Robusta
11    Robusta
12    Robusta
13    Robusta
14    Robusta
15    Robusta
16    Robusta
17    Robusta
18    Robusta
19    Robusta
20    Robusta
21    Robusta
22    Robusta
23    Robusta
24    Robusta
25    Robusta
26    Robusta
27    Robusta
28    Robusta
Name: Species, dtype: object
```

> **Important**
>
> We did not do this step in class

We can pick out rows with `loc`

```
coffee_df.loc[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.Int64HashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 0

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1867/2325123020.py in <module>
----> 1 coffee_df.loc[0]

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexing.py in __getitem__(self, key)
    929
    930             maybe_callable = com.apply_if_callable(key, self.obj)
--> 931             return self._getitem_axis(maybe_callable, axis=axis)
    932
    933     def _is_scalar_access(self, key: tuple):

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexing.py in _getitem_axis(self, key, axis)
   1162         # fall thru to straight lookup
   1163         self._validate_key(key, axis)
-> 1164         return self._get_label(key, axis=axis)
   1165
   1166     def _get_slice_axis(self, slice_obj: slice, axis: int):

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexing.py in _get_label(self, label, axis)
   1111     def _get_label(self, label, axis: int):
   1112         # GH#5667 this will fail if the label is not present in the axis.
-> 1113         return self.obj.xs(label, axis=axis)
   1114
   1115     def _handle_lowerdim_multi_index_axis0(self, tup: tuple):

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/generic.py in xs(self, key, axis, level, drop_level)
   3774                 raise TypeError(f"Expected label or tuple of labels, got
{key}") from e
   3775         else:
-> 3776             loc = index.get_loc(key)
   3777
   3778             if isinstance(loc, np.ndarray):

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 0
```

4.3. Reading data from websites

We'll first read from the course website.

> **Note**
>
> This is our first bit of web scraping! We will do more, but for very structured data it can be this easy

```
comm_url =
'https://rhodyprog4ds.github.io/BrownFall22/syllabus/communication.html#'
```

So far, we've read data in from a .csv file with `pd.read_csv` and created a DataFrame with the constructor `pd.DataFrame` using a dictionary. Pandas provides many interfaces for reading in data. They're described on the Pandas IO page.

We can use the `read_html` method to read from this page. We know that it has multiple tables on the page, and from the help, we know that it will return a list of DataFrames.

```
df_list = pd.read_html(comm_url)
```

We can also verify what it returns

```
type(df_list)
```

```
list
```

We can index with `[]` to pick one item from the list and verify that it is a DataFrame.

```
type(df_list[0])
```

```
pandas.core.frame.DataFrame
```

> **Note**
>
> Using the documentation for a library (and the base language) is totally expected and normal part of programming. That's what you should use as your primary source for questions in this class. Other sources can become outdated pretty quickly as the language changes, but most of the libraries we'll use have processes in place to ensure that their own documentation gets updated at the same time the code does.

> **Warning**
>
> If you use other sources and get advised to solutions that are deprecated you may not earn achievements for that work.

4.4. Pythonic Loops

In Python, loops do not require an iterator variable. It has an interable object and a loop variable.

```
for loop_variable in iterable_object:
    # loop body
```

the `loop_variable` takes on the value of each item in the `iterable_object` each time it goes through, in order. Writing loops this way makes them more compact and more readable, this is more like English. For example:

```
name = 'sarah'
for letter in name:
    print(letter.upper())
```

```
S
A
R
A
H
```

It is best to name variables so that the loop variable makes sense as an item from the iterable. For example, names have letters in them, and an item in `df_list` makes sense as `df`.

```
for df in df_list:
    print(df.shape)
```

```
(6, 4)
(6, 4)
(1, 3)
(3, 3)
(2, 3)
```

4.5. Types Solution

> **Warning**
>
> I am using bad variable names here `a`, `b` ,... because these are only as options for a question and we will not use them again

```
a = [char for char in 'abcde']
a
```

This is called a list comprehension. It allows you to build a list using a for loop all in one step.

```
['a', 'b', 'c', 'd', 'e']
```

```
type(a)
```

```
list
```

```
b = {char:i for i, char in enumerate('abcde')}
b
```

This is called a dictionary comprehension. It allows you to build a dictionary using a for loop all in one step.

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

```
type(b)
```

```
dict
```

```
c = ('a','b','c','d','e')
c
```

```
('a', 'b', 'c', 'd', 'e')
```

```
type(c)
```

```
tuple
```

```
d = 'a b c d e'.split('')
d
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_1867/2343856410.py in <module>
----> 1 d = 'a b c d e'.split('')
      2 d

ValueError: empty separator
```

```
type(d)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1867/2049116839.py in <module>
----> 1 type(d)

NameError: name 'd' is not defined
```

4.6. Questions After Class

4.6.1. what is a dictionary in python?

a dictionary is a datatype from base python that stores key, value pairs.

For example

```
prof_info = {'first':'Sarah', 'last':'Brown', 'title':'Dr.'}
prof_info
```

```
{'first': 'Sarah', 'last': 'Brown', 'title': 'Dr.'}
```

We can use the keys to index in and get the values out

```
prof_info['title']
```

```
'Dr.'
```

Even though we will mostly use DataFrame, dictionaries and other base python types are important. Dictionaries are very powerful they can hold whole functions in them. For example, the Python language does not have a switch case (which can be used for handling many if/else cases) but instead dictionaries can be used for that.

> **Further Reading**
>
> You can read more about the details of data types in Pandas in the documentation

4.6.2. How to see unique values in a column

We will get to this soon! We got the first part, picking out a single column to look at, we will see the method for that probably on Monday, but maybe on Friday.

5. Iterables and Indexing

5.1. Logistics

5.1.1. Assignment

On Assignment 2, there's a script to run to prepare it for grading.

> **Important**
>
> I missed a file in the template, so please create a file called requirements.txt that has the following contents.
>
> ```
> jupyter-book
> pyppeteer
> pandas
> jupytext
> ```

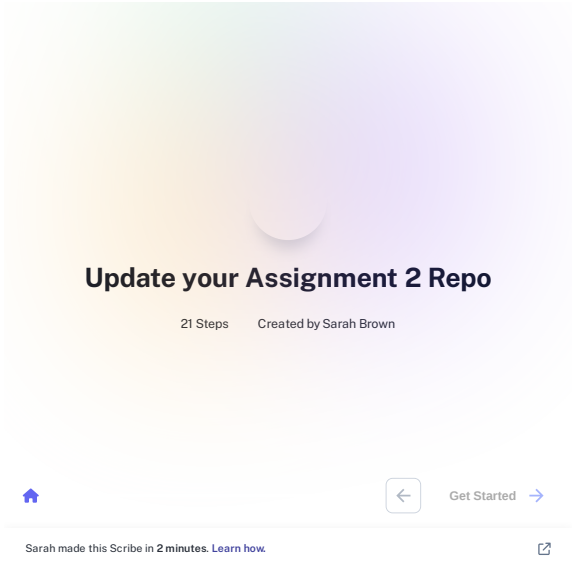
Correct script file (for .githu/workflows/submit.yml)

```
name: Prepare and Submit
on:
  workflow_dispatch

jobs:
  generatereport:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    # Install dependencies
    - name: Set up Python 3.9
      uses: actions/setup-python@v1
      with:
        python-version: 3.9

    # install dependencies
    - name: dependencies
      run: pip install -r requirements.txt

    # generate the report
    - name: Convert notebooks to md for reading
      run: |
        jupytext --to myst *.ipynb
        jupytext --to myst */*.ipynb

    # compile the pdf
    - name: Build a basic html to pdf
      run: |
        jupyter-book build *.ipynb --builder pdfhtml

    # start a pull request for it
    - name: Create Pull Request
      uses: peter-evans/create-pull-request@v4
      with:
        commit-message: 'convert to md'
        draft: false
        branch: published
        base: main
        title: Submission
```

Add this to `.gitignore`

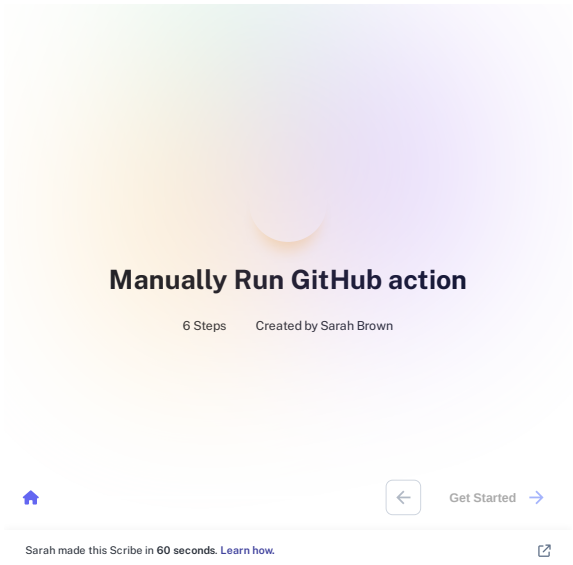
```
_build/
```

How to apply these updates:



full view

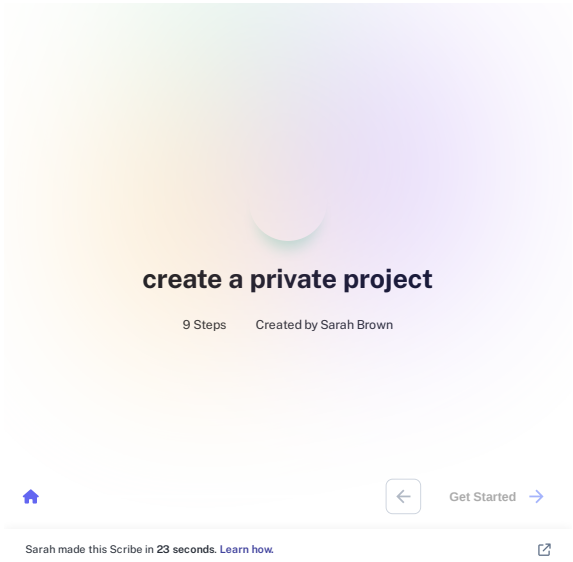
To run the script



see instructions in a full page

5.1.2. Grade Tracking

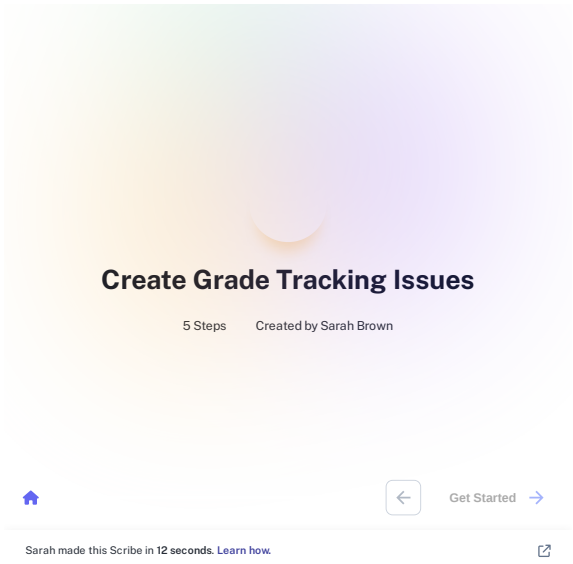
Create a Project board



Then create your repo and run the manual workflow on that actions tab to create issues:

Note

Are these scribe screenshots helpful? do you want more of them?



5.2. Lists of lists

```
import pandas as pd
```

Recall how indexing works with negagive numbers

```
topics = ['what is data science', 'jupyter', 'conditional','functions', 'lists',
'dictionaries','pandas' ]
```

```
topics[-1]
```

```
'pandas'
```

A list we built together:

```
# You can remain anonymous (this page & the notes will be fully public)
# by attributing it to a celebrity or psuedonym, but include *some* sort of
attribution
sentence_list = [
"The class is just starting to feel settled for me. - Dr. Brown",
"",
"Hello, I like sushi! - ",
"Why squared  aka the mask - is a computer science student."
"Data science is fun",
"Hello my fellow gaymers - Sun Tzu",
"Soccer is a sport -Obama",
"Hello, I love pizza - Bear",
"This class is CSC/DSP 310. - Student",
"It is 2:21pm -",
"Pizza conquers all- Beetlejuice"
"",
"ayyy whaddup wit it - frankie",
"This is a sentence - George W Bush",
"Steam is the best place to play videogames change my mind. - Todd Howard",
"This is a hello -",
"Hello how are you -",
"The monkey likes bananas. - A banana",
"",
"Just type a random sentence - Rosa Parks",
"",
"I love CSC. - Everyone",
"",
"The quick brown fox jumps over the lazy dog - Brendan Chadwick",
"I like computers - David",
"",
"The fitness gram pacer test is a multi aerobic capacity test - Matt 3",
"Sally sells seashells by the seashore. - Narrator",
"I would like to take a nap. - Tom Cruise,"
]
```

We can confirm this is a list

```
type(sentence_list)
```

```
list
```

We can use a list comprehension to remove the empty ones.

```
sentences_clean = [s for s in sentence_list if len(s)>1]
sentences_clean
```

```
['The class is just starting to feel settled for me. - Dr. Brown',
 'Hello, I like sushi! - ',
 'Why squared  aka the mask - is a computer science student.Data science is fun',
 'Hello my fellow gaymers - Sun Tzu',
 'Soccer is a sport -Obama',
 'Hello, I love pizza - Bear',
 'This class is CSC/DSP 310. - Student',
 'It is 2:21pm -',
 'Pizza conquers all- Beetlejuice',
 'ayyy whaddup wit it - frankie',
 'This is a sentence - George W Bush',
 'Steam is the best place to play videogames change my mind. - Todd Howard',
 'This is a hello -',
 'Hello how are you -',
 'The monkey likes bananas. - A banana',
 'Just type a random sentence - Rosa Parks',
 'I love CSC. - Everyone',
 'The quick brown fox jumps over the lazy dog - Brendan Chadwick',
 'I like computers - David',
 'The fitness gram pacer test is a multi aerobic capacity test - Matt 3',
 'Sally sells seashells by the seashore. - Narrator',
 'I would like to take a nap. - Tom Cruise,']
```

We can use the `split` method on a string to make a list of lists

```
sentence_data = [s.split('-') for s in sentences_clean]
sentence_data
```

```
[['The class is just starting to feel settled for me. ', ' Dr. Brown'],
 ['Hello, I like sushi! ', ' '],
 ['Why squared  aka the mask ',
  ' is a computer science student.Data science is fun'],
 ['Hello my fellow gaymers ', ' Sun Tzu'],
 ['Soccer is a sport ', 'Obama'],
 ['Hello, I love pizza ', ' Bear'],
 ['This class is CSC/DSP 310. ', ' Student'],
 ['It is 2:21pm ', ''],
 ['Pizza conquers all', ' Beetlejuice'],
 ['ayyy whaddup wit it ', ' frankie'],
 ['This is a sentence ', ' George W Bush'],
 ['Steam is the best place to play videogames change my mind. ',
  ' Todd Howard'],
 ['This is a hello ', ''],
 ['Hello how are you ', ''],
 ['The monkey likes bananas. ', ' A banana'],
 ['Just type a random sentence ', ' Rosa Parks'],
 ['I love CSC. ', ' Everyone'],
 ['The quick brown fox jumps over the lazy dog ', ' Brendan Chadwick'],
 ['I like computers ', ' David'],
 ['The fitness gram pacer test is a multi aerobic capacity test ', ' Matt 3'],
 ['Sally sells seashells by the seashore. ', ' Narrator'],
 ['I would like to take a nap. ', ' Tom Cruise,']]
```

then we can use the DataFrame Constructor)

```
sentence_df = pd.DataFrame(data= sentence_data, columns=
['sentence','attribution'])
```

We can use head by default for 5

```
sentence_df.head()
```

	sentence	attribution
0	The class is just starting to feel settled for...	Dr. Brown
1	Hello, I like sushi!	
2	Why squared aka the mask	is a computer science student.Data science is...
3	Hello my fellow gaymers	Sun Tzu
4	Soccer is a sport	Obama

or pass a nubmer to get a different number of rows

```
sentence_df.head(3)
```

	sentence	attribution
0	The class is just starting to feel settled for...	Dr. Brown
1	Hello, I like sushi!	
2	Why squared aka the mask	is a computer science student.Data science is...

the `loc` property can index on rows or columns, but is rows by deault

```
sentence_df.loc[3]
```

```
sentence       Hello my fellow gaymers
attribution                   Sun Tzu
Name: 3, dtype: object
```

We can select a range, like in base python with a colon

```
sentence_df.loc[3:5]
```

	sentence	attribution
3	Hello my fellow gaymers	Sun Tzu
4	Soccer is a sport	Obama
5	Hello, I love pizza	Bear

5.3. Loading a json

We can load a json using the `read_json` method the same way we used the `read_csv`

```
rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
```

```
pd.read_json(rhodyprog4ds_gh_events_url)
```

	id	type	actor	repo	payload	public	created_at	org
0	24288681299	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11169445176, 'size': 2, 'distinct_...	True	2022-09-29 02:26:45+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
1	24288620658	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11169413015, 'size': 1, 'distinct_...	True	2022-09-29 02:21:13+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
2	24288587636	IssuesEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'closed', 'issue': {'url': 'https:/...	True	2022-09-29 02:18:13+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
3	24288587532	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11169396090, 'size': 1, 'distinct_...	True	2022-09-29 02:18:13+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
4	24281758037	IssuesEvent	{'id': 71090172, 'login': 'SabrinaSays', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'opened', 'issue': {'url': 'https:/...	True	2022-09-28 18:03:49+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
5	24281272990	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11165686448, 'size': 1, 'distinct_...	True	2022-09-28 17:37:53+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
6	24281208363	PushEvent	{'id': 90274412, 'login': 'aHill024', 'display...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11165655037, 'size': 1, 'distinct_...	True	2022-09-28 17:34:23+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
7	24281038794	PushEvent	{'id': 90274412, 'login': 'aHill024', 'display...	{'id': 400656699, 'name': 'rhodyprog4ds/rhodyd...	{'push_id': 11165572700, 'size': 1, 'distinct_...	True	2022-09-28 17:25:28+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
8	24237721326	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11144288905, 'size': 1, 'distinct_...	True	2022-09-27 02:02:58+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
9	24237700507	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-09-27 02:01:04+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
10	24237693940	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c9', 'ref_type': 'tag', 'master_branc...	True	2022-09-27 02:00:27+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
11	24237687041	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11144270025, 'size': 1, 'distinct_...	True	2022-09-27 01:59:53+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
12	24237648960	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11144249777, 'size': 1, 'distinct_...	True	2022-09-27 01:56:25+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
13	24237611833	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11144229958, 'size': 1, 'distinct_...	True	2022-09-27 01:52:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
14	24195811138	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120860892, 'size': 1, 'distinct_...	True	2022-09-23 23:20:22+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
15	24195801542	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-09-23 23:19:12+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
16	24195786680	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c8', 'ref_type': 'tag', 'master_branc...	True	2022-09-23 23:17:23+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
17	24195781471	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120844321, 'size': 9, 'distinct_...	True	2022-09-23 23:16:48+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
18	24195781397	PullRequestEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'closed', 'number': 15, 'pull_reque...	True	2022-09-23 23:16:47+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
19	24195779374	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120843249, 'size': 1, 'distinct_...	True	2022-09-23 23:16:33+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
20	24195773150	PullRequestEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'opened', 'number': 15, 'pull_reque...	True	2022-09-23 23:15:49+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...

	id	type	actor	repo	payload	public	created_at	org
21	24195772674	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120839590, 'size': 6, 'distinct_...	True	2022-09-23 23:15:45+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
22	24195097479	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120515220, 'size': 1, 'distinct_...	True	2022-09-23 22:22:31+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
23	24195064154	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11120499394, 'size': 2, 'distinct_...	True	2022-09-23 22:20:03+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
24	24195014440	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'noted923', 'ref_type': 'branch', 'mas...	True	2022-09-23 22:16:08+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
25	24192818852	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-09-23 19:30:32+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
26	24192727151	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11119317177, 'size': 1, 'distinct_...	True	2022-09-23 19:24:23+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
27	24192704458	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'a3', 'ref_type': 'tag', 'master_branc...	True	2022-09-23 19:22:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
28	24192688708	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11119297628, 'size': 1, 'distinct_...	True	2022-09-23 19:21:46+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
29	24191110266	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11118507916, 'size': 1, 'distinct_...	True	2022-09-23 17:40:27+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...

5.4. Working with your repo offline

> **Warning**
>
> This was not done in class and is optional

5.4.1. Authenticate on Mac

On macOS install GitHub CLI

```
gh auth login
```

use defualts and choose to log in via browser.

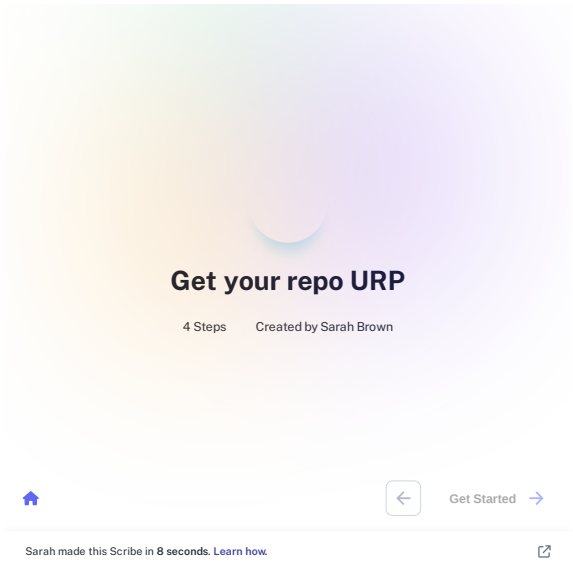
5.4.2. Authenticate on Windows

On winows install GitBash

Then try to do the clone step and GitBash will help you authenticate

5.4.3. Work offline

Get your repo URL:



cd to where you want to save

```
cd prog4ds
git clone https://github.com/rhodyprog4ds/02-loading-data-brownsarahm.git
```

work in the new folder that creates

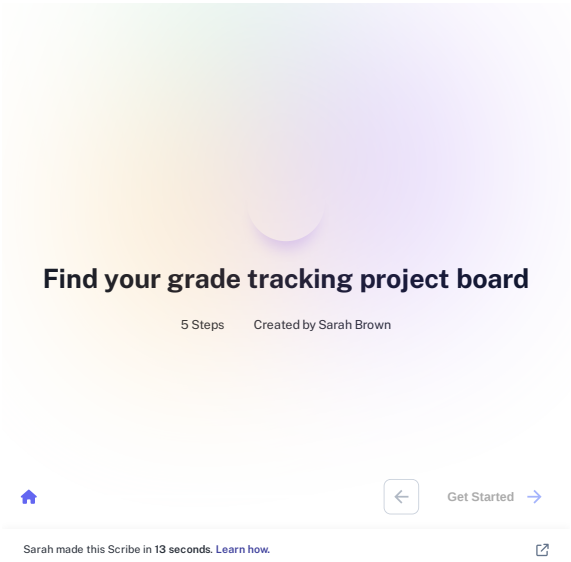
When you want to Save

```
git add .
git commit -m 'describe the work you did'
git push
```

5.5. Questions After Class

5.5.1. Logistics

5.5.1.1. where do we find the grading page?



5.5.2. Assignemnt

5.5.2.1. what are the keys needed on the dictionaries for the assignment?

See the `datasets.py` file in the template repo

5.5.2.2. do we have to accept assignment 2 anywhere and if so how

Yes on the assignment page. The link says “accept the assignment”

5.5.2.3. For the purposes of the assignment should we download it locally to work with our notebooks?

Read the instructions carefully on the assignment. It tells you exactly what to do.

5.5.3. Content

5.5.3.1. How do you locate a specific row and column from a dataframe?

`.loc` accepts both, using a comma to separate. The docs for `loc` have lots of examples.

5.5.3.2. with data sets if there is an error with formatting and we can modify the original how would we fix it

Download to a copy where you can edit.

5.5.3.3. Can you use .loc to pull out multiple rows that aren’t next to each other. For example, if I wanted to view rows 3, 8 and 12

Yes, to select multiple nonconsecutive, you pass a list. The docs for `loc` have lots of examples.

5.5.3.4. how can we iterate through dictionaries

dictionaries have a `.items()` method that pops off tuples of th key and value.

> **Warning**
>
> the assignment does not ask you to iterate through a dictionary object, but over a list of dictionaries

5.5.3.5. What is the main differnce between JSON and csv files; does one allocate more memory / store larger sets?

The main difference is the structure. JSON can hold nested data. For example look at the GitHub data that we read in in class.

5.5.3.6. what exactly does json mean / do

json is a data file format. It is an acronym for JavaScript Object Notation. It’s a popular format for internet content.

5.5.3.7. are nested lists the only way to create DataFrames in python

nested lists are not the only way to create pandas DataFrames, you can also do that from a Dictionary. see in the docs for the constructor.

5.5.3.8. How do nested loops work in the jupyter notebook

Python constructs other than display items work just as they do in any other interpretter in a jupyter notebook. The list comprehension that we saw today also works in base Python. You can nest list comphrehenions in different ways depending on your goal.

6. Getting Started with Exploratory Data Analysis

Now we get to start actual data science!

Our goal this week is to explore the actual data, the values, in a dataset to get a basic idea of what is in the data.

Summarizing and Visualizing Data are **very** important

- People cannot interpret high dimensional or large samples quickly
- Important in EDA to help you make decisions about the rest of your analysis
- Important in how you report your results
- Summaries are similar calculations to performance metrics we will see later
- visualizations are often essential in debugging models

THEREFORE

- You have a lot of chances to earn summarize and visualize
- we will be picky when we assess if you earned them or not

```
import pandas as pd
```

6.1. Staying Organized

- See the new File structure section.
- Be sure to accept assignments and close the feedback PR if you will not work on them

6.2. Loading Data and Examining Structure

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
```

So far, we've loaded data in a few different ways and then we've examined DataFrames as a data structure, looking at what different attributes they have and what some of the methods are, and how to get data into them.

```
coffee_df.head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expirat
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2	June 2 2
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2	Octo 31st, 2

2 rows × 43 columns

From here we can see a few sample values of most of the column and we can be sure that the data loaded correctly.

> **Try it Yourself**
>
> What other tools have we learned to examine a DataFrame and when might you use them?

We can also get more structural information with the **`info`** method.

More information on this can also be found in the dtypes attribute. Including that the type is the most general if there are multiple types in the column.

```
coffee_df.info()
```

> **Hint**
>
> There is also a related **`select_dtypes`** method.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28 entries, 1 to 28
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Species                28 non-null     object
 1   Owner                  28 non-null     object
 2   Country.of.Origin      28 non-null     object
 3   Farm.Name              25 non-null     object
 4   Lot.Number             6 non-null      object
 5   Mill                   20 non-null     object
 6   ICO.Number             17 non-null     object
 7   Company                28 non-null     object
 8   Altitude               25 non-null     object
 9   Region                 26 non-null     object
 10  Producer               26 non-null     object
 11  Number.of.Bags         28 non-null     int64
 12  Bag.Weight             28 non-null     object
 13  In.Country.Partner     28 non-null     object
 14  Harvest.Year           28 non-null     int64
 15  Grading.Date           28 non-null     object
 16  Owner.1                28 non-null     object
 17  Variety                3 non-null      object
 18  Processing.Method      10 non-null     object
 19  Fragrance...Aroma      28 non-null     float64
 20  Flavor                 28 non-null     float64
 21  Aftertaste             28 non-null     float64
 22  Salt...Acid            28 non-null     float64
 23  Bitter...Sweet         28 non-null     float64
 24  Mouthfeel              28 non-null     float64
 25  Uniform.Cup            28 non-null     float64
 26  Clean.Cup              28 non-null     float64
 27  Balance                28 non-null     float64
 28  Cupper.Points          28 non-null     float64
 29  Total.Cup.Points       28 non-null     float64
 30  Moisture               28 non-null     float64
 31  Category.One.Defects   28 non-null     int64
 32  Quakers                28 non-null     int64
 33  Color                  26 non-null     object
 34  Category.Two.Defects   28 non-null     int64
 35  Expiration             28 non-null     object
 36  Certification.Body     28 non-null     object
 37  Certification.Address  28 non-null     object
 38  Certification.Contact  28 non-null     object
 39  unit_of_measurement    28 non-null     object
 40  altitude_low_meters    25 non-null     float64
 41  altitude_high_meters   25 non-null     float64
 42  altitude_mean_meters   25 non-null     float64
dtypes: float64(15), int64(5), object(23)
memory usage: 9.6+ KB
```

6.3. Summary Statistics

Now, we can actually start to analyze the data itself.

The **`describe`** method provides us with a set of summary statistics that broadly describe the data overall.

```
coffee_df.describe()
```

	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste	Salt...Acid	Bitter...Sweet	Mouthfeel	Uniform.Cup	Clean.Cup	Balance	Cupper.Points
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000
mean	168.000000	2013.964286	7.702500	7.630714	7.559643	7.657143	7.675714	7.506786	9.904286	9.928214	7.541786	7.761429
std	143.226317	1.346660	0.296156	0.303656	0.342469	0.261773	0.317063	0.725152	0.238753	0.211030	0.526076	0.330507
min	1.000000	2012.000000	6.750000	6.670000	6.500000	6.830000	6.670000	5.080000	9.330000	9.330000	5.250000	6.920000
25%	1.000000	2013.000000	7.580000	7.560000	7.397500	7.560000	7.580000	7.500000	10.000000	10.000000	7.500000	7.580000
50%	170.000000	2014.000000	7.670000	7.710000	7.670000	7.710000	7.750000	7.670000	10.000000	10.000000	7.670000	7.830000
75%	320.000000	2015.000000	7.920000	7.830000	7.770000	7.830000	7.830000	7.830000	10.000000	10.000000	7.830000	7.920000
max	320.000000	2017.000000	8.330000	8.080000	7.920000	8.000000	8.420000	8.250000	10.000000	10.000000	8.000000	8.580000

From this, we can draw several conclusions. FOr example straightforward ones like:

- the smallest number of bags rated is 1 and at least 25% of the coffees rates only had 1 bag
- the first ratings included were 2012 and last in 2017 (min & max)
- the mean Mouthfeel was 7.5
- Category One defects are not very common (the 75th% is 0)

Or more nuanced ones that compare across variables like

- the raters scored coffee higher on Uniformity.Cup and Clean.Cup than other scores (mean score; only on the ones that seem to have a scale of up to 8/10)
- the coffee varied more in Mouthfeel and Balance that most other scores (the std; only on the ones that seem to have a scale of of up to 8/10)
- there are 3 ratings with no altitude (count of other variables is 28; alt is 25

And these all give us a sense of the values and the distribution or spread fo the data in each column.

We can use the descriptive statistics on individual columns as well.

```
coffee_df['Balance'].describe()
```

```
count    28.000000
mean      7.541786
std       0.526076
min       5.250000
25%       7.500000
50%       7.670000
75%       7.830000
max       8.000000
Name: Balance, dtype: float64
```

To dig in on what the quantiles really mean, we can compute one manually.

First, we sort the data, then for the 25%, we select the point in index 6 because becaues there are 28 values.

```
balance_sorted = coffee_df['Balance'].sort_values().values
```

What value of `x` to pick out 25%

```
x = 6
balance_sorted[x]
```

```
7.5
```

We can also extract each of the statistics that the `describe` method calculates individually, by name. The quantiles are tricky, we cannot just `.25%()` to get the 25% percentile, we have to use the `quantile` method and pass it a value between 0 and 1.

```
coffee_df['Flavor'].quantile(.8)
```

```
7.83
```

Calculate the mean of `Aftertaste`

```
coffee_df['Aftertaste'].mean()
```

```
7.559642857142856
```

6.4. Describing Nonnumerical Variables

There are different columns in the describe than the the whole dataset:

```
coffee_df.columns
```

```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year',
       'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method',
       'Fragrance...Aroma', 'Flavor', 'Aftertaste', 'Salt...Acid',
       'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup', 'Clean.Cup', 'Balance',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Color', 'Category.Two.Defects', 'Expiration',
       'Certification.Body', 'Certification.Address', 'Certification.Contact',
       'unit_of_measurement', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

```
coffee_df.describe().columns
```

```
Index(['Number.of.Bags', 'Harvest.Year', 'Fragrance...Aroma', 'Flavor',
       'Aftertaste', 'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel',
       'Uniform.Cup', 'Clean.Cup', 'Balance', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers',
       'Category.Two.Defects', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

We can get the prevalence of each one with `value_counts`

```
coffee_df['Color'].value_counts()
```

```
Green           20
Blue-Green       3
Bluish-Green     2
None             1
Name: Color, dtype: int64
```

Try it Yourself

Note `value_counts` does not count the `NaN` values, but `count` counts all of the not missing values and the shape of the DataFrame is the total number of rows. How can you get the number of missing Colors?

Describe only operates on the numerical columns, but we might want to know about the others. We can get the number of each value with `value_counts`

```
coffee_df['Country.of.Origin'].value_counts()
```

```
India            13
Uganda           10
United States     2
Ecuador           2
Vietnam           1
Name: Country.of.Origin, dtype: int64
```

We can get the name of the most common country out of this Series using `idmax`

```
coffee_df['Country.of.Origin'].value_counts().idxmax()
```

```
'India'
```

Or see only how many different values with the related:

```
coffee_df['Country.of.Origin'].nunique()
```

```
5
```

We can also use the mode function, which works on both numerical or nonnumerical

```
coffee_df['Country.of.Origin'].mode()
```

```
0    India
dtype: object
```

further reading

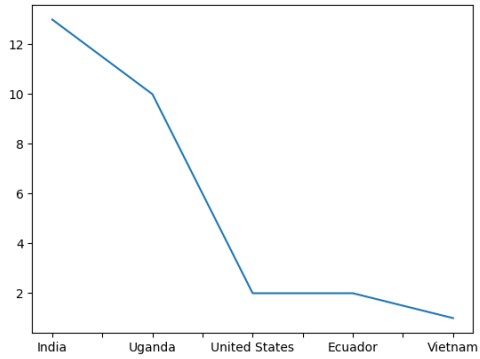
On the documentation page for describe the "See Also" shows the links to the documentation of most of the individual functions. This is a good way to learn about other things, or find something when you are not quite sure what it would be named. Go to a function thats similar to what you want and then look at the related functions.

6.5. Basic Plotting

Pandas give us basic plotting capability built right into DataFrames.

```
coffee_df['Country.of.Origin'].value_counts().plot()
```

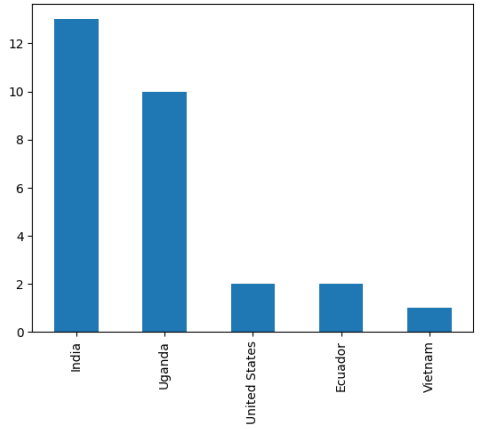
```
<AxesSubplot:>
```



It defaults to a line graph, which is not very informative in this case, so we can use the `kind` parameter to change it to a bar graph.

```
coffee_df['Country.of.Origin'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



6.6. Matching Questions to Summary Statistics

When we brainstorm more questions that we can answer with summary statistics or that we might want to ask of this data, the ones that come to mind are often actually dependent on two variables. For example are two scores correlated? or what country has the highest rated coffe on `Flavor`?

We'll come back to more detailed questions like this on Friday, but to start we can look at how numerical variables vary by categorical (nonnumerical) variables by grouping the data.

Above we saw which country had the most ratings (remember one row is one rating), but what if we wanted to see the mean number of bags per country?

```
coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].mean()
```

```
Country.of.Origin
Ecuador            1.000000
India            230.076923
Uganda           160.900000
United States     50.500000
Vietnam            1.000000
Name: Number.of.Bags, dtype: float64
```

> **Important**
>
> This data is only about coffee that was rated by a particular agency it is not economic data, so we cannot, for example conclude which country *produces* the amount of data. If we had economic dataset, a `Number.of.Bags` columns's mean would tell us exactly that, but the context of the dataset defines what a row means and therefore how we can interpret the **every single statistic** we calculate.

6.7. Questions after class

6.7.1. Can arrays be used in Jupyter Notebook?

Jupyter runs a fully powered python interpreter, so all python can work inside it.

6.7.1.1. why did coffee_df['Country.of.Origin'].max() say vietnam?

That applies a the `max` function to the `'Country.of.Origin'` column, without counting how many times the values occur.

6.7.1.2. Why, in the groupby function the first column is in between parenthesis nd the second one between brackets?

The inside parenthesis is the parameter of the groupby function, we could instead do it this way:

```
coffee_grouped = coffee_df.groupby('Country.of.Origin')
```

Then we can use `coffee_grouped` for different things

```
coffee_grouped.describe()
```

	Number.of.Bags								Harvest.Year		...	altitude_high_meters		altitude_mean_meters			
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	mir
Country.of.Origin																	
Ecuador	2.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0	2.0	2016.000000	...	40.00	40.0	1.0	40.000000	NaN	4
India	13.0	230.076923	110.280296	1.0	140.00	300.0	320.00	320.0	13.0	2014.384615	...	1500.00	3170.0	12.0	1421.666667	1021.022957	75
Uganda	10.0	160.900000	163.690392	1.0	2.25	160.0	320.00	320.0	10.0	2013.300000	...	1488.00	1745.0	10.0	1354.500000	220.042546	109
United States	2.0	50.500000	70.003571	1.0	25.75	50.5	75.25	100.0	2.0	2013.000000	...	2448.75	3000.0	2.0	1897.500000	1559.170453	79
Vietnam	1.0	1.000000	NaN	1.0	1.00	1.0	1.00	1.0	1.0	2013.000000	...	NaN	NaN	0.0	NaN	NaN	N

5 rows × 160 columns

or as we did before

```
coffee_grouped['Number.of.Bags'].mean()
```

```
Country.of.Origin
Ecuador             1.000000
India             230.076923
Uganda            160.900000
United States      50.500000
Vietnam             1.000000
Name: Number.of.Bags, dtype: float64
```

The second one is to index (see last week's notes for more on indexing) the DataFrame and pick out one column. It makes the DataFrame have fewer columns before applying mean to the whole object.

We could make a smaller DataFrame first, then group, then mean

```
coffee_country_bags_df= coffee_df[['Number.of.Bags','Country.of.Origin']]
coffee_country_bags_df.head()
```

	Number.of.Bags	Country.of.Origin
1	300	Uganda
2	320	India
3	300	India
4	320	Uganda
5	1	Uganda

There are two sets of square brackets because putting a comma it would try to index along rows with one and columns with the other, to select multiple in one dimension you need to pass a list.

```
type(['Number.of.Bags','Country.of.Origin'])
```

```
list
```

If you do not put square `[]` or curly `{}` brackets around items, Python implicitly treats them as if there are parenthesis`()`

```
type(('Number.of.Bags','Country.of.Origin'))
```

```
tuple
```

and tuples are treated separately .

```
coffee_country_bags_df.groupby('Country.of.Origin').mean()
```

	Number.of.Bags
Country.of.Origin	
Ecuador	1.000000
India	230.076923
Uganda	160.900000
United States	50.500000
Vietnam	1.000000

6.7.1.3. A quick recap of how .info() counts non-null and dtype for each column would be great. Thanks!

above

6.7.1.4. Can you plot a graph that uses two columns in a dataset?

Yes, we will see that on Wednesday.

6.7.1.5. Are there any other data types as important as dataframes in pandas?

DataFrames are the main type provided by pandas, there are also Series which is conceptually a single column, groupby objects which is basically a list of (DataFrame,label) tuples, and some indexing, but all of these exist to support creating and operating on DataFrames.

6.7.1.6. Will we be utilizing any other packages /libraries mentioned than the ones for visualizing data that we talked about today?

We will use pandas for basic plots and seaborn for more advanced plots. There are other libraries for visualization like plotly for interactive plots, but those are more advanced than we need or for specialized use cases. However, the way these type of libraries are designed, is that the special use case ones are for when the basic ones do not do what you need and they often have common patterns.

6.7.1.7. When displaying two categories like we did at the end could you measure a different thing for each category? Like count for one and mean for the other?

Yes! We will likely not do this in class, but it is a small extension from what we have covered and uses the pandas aggregrate method.

> **Hint**
>
> This is expected for visualize level 3

6.7.1.8. How in depth will we go with graphing data and looking at it systemically to find patterns?

At some level, that is what we sill do for the rest of the semester. We will not cover everything there is to know about graphing, there are whole courses on data visualization. Most of the pattern-finding we will do is machine learning.

6.7.2. Assignment

6.7.2.1. For assignment 2, how do I find appropriate datasets online?

I collated the ones that I recommend on the datasets page. Then find something that is of interest to you.

6.7.2.2. How do you import a .py file from GitHub? Is there a way to do that without downloading it locally?

It does need to be local in order to import it as a module.

Ideally, you will work with all of your assignment repos locally either following the instructions in the notes from last class

7. Visualization

```
import pandas as pd
```

7.1. A note on viewing output

When we create a variable and then put that on the last line of a cell, jupyter *displays* it.

```
name = 'sarah'
name
```

```
'sarah'
```

How it displays it depends on the type

```
type(name)
```

```
str
```

For a string, it uses `print`

```
print(name)
```

```
sarah
```

so this and the one above look the same. For objects that have a `_repr_html_` method, jupyter uses that, and then your brownser uses html to render the object in a more visually appealing way.

7.2. Review from Monday

We will load the robusta data briefly again.

```
robusta_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

Is the robusta coffee's Mouthfeel or Aftertaste rated more consistently

```
robusta_df = pd.read_csv(robusta_data_url)
robusta_df.describe()
```

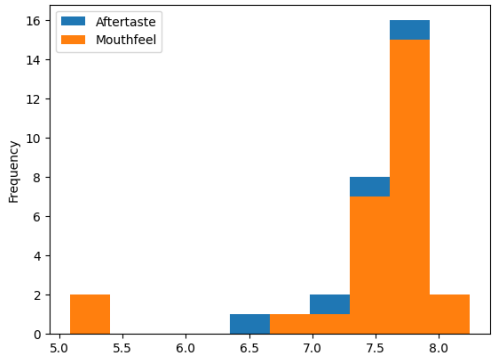
	Unnamed: 0	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste	Salt...Acid	Bitter...Sweet	Mouthfeel	Uniform.Cup	...	Balance	Cupper.Poin
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	...	28.000000	28.0000
mean	14.500000	168.000000	2013.964286	7.702500	7.630714	7.559643	7.657143	7.675714	7.506786	9.904286	...	7.541786	7.7614
std	8.225975	143.226317	1.346660	0.296156	0.303656	0.342469	0.261773	0.317063	0.725152	0.238753	...	0.526076	0.3305
min	1.000000	1.000000	2012.000000	6.750000	6.670000	6.500000	6.830000	6.670000	5.080000	9.330000	...	5.250000	6.9200
25%	7.750000	1.000000	2013.000000	7.580000	7.560000	7.397500	7.560000	7.580000	7.500000	10.000000	...	7.500000	7.5800
50%	14.500000	170.000000	2014.000000	7.670000	7.710000	7.670000	7.710000	7.750000	7.670000	10.000000	...	7.670000	7.8300
75%	21.250000	320.000000	2015.000000	7.920000	7.830000	7.770000	7.830000	7.830000	7.830000	10.000000	...	7.830000	7.9200
max	28.000000	320.000000	2017.000000	8.330000	8.080000	7.920000	8.000000	8.420000	8.250000	10.000000	...	8.000000	8.5800

8 rows × 21 columns

from the lower `std` we can see that Aftertaste is more consistently rated.

```
robusta_df[['Aftertaste','Mouthfeel']].plot(kind='hist')
```

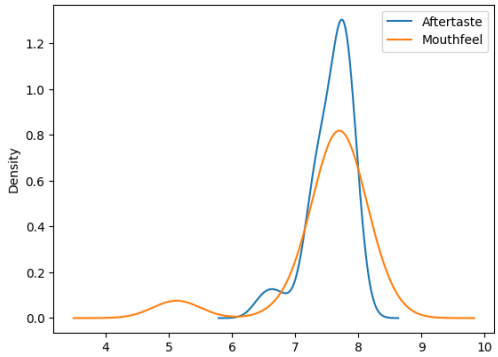
```
<AxesSubplot:ylabel='Frequency'>
```



We can change the kind, for example to a Kernel Density Estimate. This approximates the distribution of the data, you can think of it rougly like a smoothed out histogram.

```
robusta_df[['Aftertaste','Mouthfeel']].plot(kind='kde')
```

```
<AxesSubplot:ylabel='Density'>
```



This version makess it more visually clear that the the Aftertaste is more consistently, but it also helps us see that that might not be the whole story. Both have a second smaller bump, so the overall std might not be the best measure.

> **Question from class**
>
> Why do we need two sets of brackets?

It tries to use them to index in multiple ways instead.

```
robusta_df['Aftertaste','Mouthfeel']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Aftertaste', 'Mouthfeel')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1934/2132958936.py in <module>
----> 1 robusta_df['Aftertaste','Mouthfeel']

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/frame.py in __getitem__(self, key)
   3456             if self.columns.nlevels > 1:
   3457                 return self._getitem_multilevel(key)
-> 3458             indexer = self.columns.get_loc(key)
   3459             if is_integer(indexer):
   3460                 indexer = [indexer]

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: ('Aftertaste', 'Mouthfeel')
```

It tries to look for a multiindex, but we do not have one so it fails. THe second square brackets, makes it a list of names to use and pandas looks for them sequentially.

7.3. Comparing two datasets

we're going to work with the arabica data today, because it's a little bigger and more interesting for plotting

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
arabica_df = pd.read_csv(arabica_data_url,index_col=0)
```

It is mostly the same columns as the robusta data

```
arabica_df.columns
```

```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year',
       'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method', 'Aroma',
       'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance', 'Uniformity',
       'Clean.Cup', 'Sweetness', 'Cupper.Points', 'Total.Cup.Points',
       'Moisture', 'Category.One.Defects', 'Quakers', 'Color',
       'Category.Two.Defects', 'Expiration', 'Certification.Body',
       'Certification.Address', 'Certification.Contact', 'unit_of_measurement',
       'altitude_low_meters', 'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

```
robusta_df.columns
```

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Fragrance...Aroma', 'Flavor', 'Aftertaste',
       'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup',
       'Clean.Cup', 'Balance', 'Cupper.Points', 'Total.Cup.Points', 'Moisture',
       'Category.One.Defects', 'Quakers', 'Color', 'Category.Two.Defects',
       'Expiration', 'Certification.Body', 'Certification.Address',
       'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
       'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

but it has a lot more rows

```
len(arabica_df) - len(robusta_df)
```

```
1283
```

```
arabica_df.shape, robusta_df.shape
```

```
((1311, 43), (28, 44))
```

7.4. Plotting in Python

- matplotlib: low level plotting tools
- seaborn: high level plotting with opinionated defaults
- ggplot: plotting based on the ggplot library in R.

Pandas and seaborn use matplotlib under the hood.

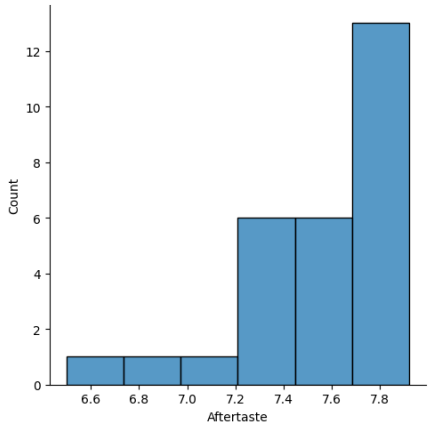
Seaborn and ggplot both assume the data is set up as a DataFrame. Getting started with seaborn is the simplest, so we'll use that.

> **Think Ahead**
>
> Learning ggplot is a way to earn level 3 for visualize

```
import seaborn as sns
```

```
sns.displot(data = robusta_df, x='Aftertaste')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fad6ff63110>
```




```
sns.displot(data = robusta_df, x='Moutfeel')
```

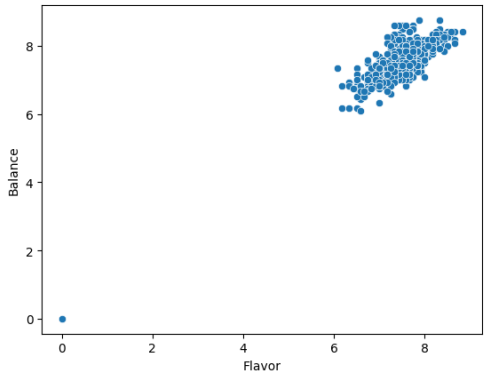
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_1934/3047069671.py in <module>
----> 1 sns.displot(data = robusta_df, x='Moutfeel')

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/seaborn/distributions.py in displot(data, x, y, hue, row, col, weights,
kind, rug, rug_kws, log_scale, legend, palette, hue_order, hue_norm, color,
col_wrap, row_order, col_order, height, aspect, facet_kws, **kwargs)
   2131     p = _DistributionFacetPlotter(
   2132         data=data,
-> 2133         variables=_DistributionFacetPlotter.get_semantics(locals())
   2134     )
   2135

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/seaborn/distributions.py in __init__(self, data, variables)
    110     ):
    111
--> 112         super().__init__(data=data, variables=variables)
    113
    114     @property

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/seaborn/_oldcore.py in __init__(self, data, variables)
    638         # information for numeric axes would be information about log
scales.
    639         self._var_ordered = {"x": False, "y": False}  # alt., used
DefaultDict
--> 640         self.assign_variables(data, variables)
    641
    642         for var, cls in self._semantic_mappings.items():

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/seaborn/_oldcore.py in assign_variables(self, data, variables)
    700             self.input_format = "long"
    701             plot_data, variables = self._assign_variables_longform(
--> 702                 data, **variables,
    703             )
    704

/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/seaborn/_oldcore.py in _assign_variables_longform(self, data, **kwargs)
    936
    937                 err = f"Could not interpret value `{val}` for parameter
`{key}`"
--> 938                 raise ValueError(err)
    939
    940             else:

ValueError: Could not interpret value `Moutfeel` for parameter `x`
```

to plot these together with seaborn we have to transform the DataFrame, so we will see that next year.

7.4.1. how are flavor and balance related?

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance')
```

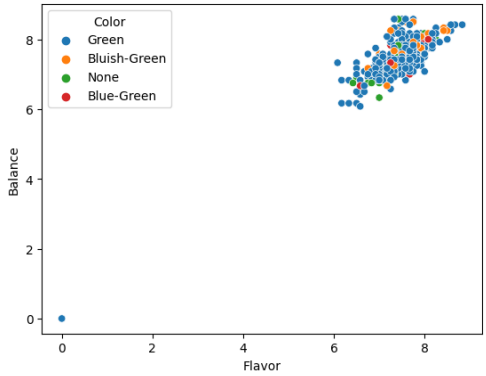
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



But now we have more power to investigate more relationships in the data.

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance',hue='Color')
```

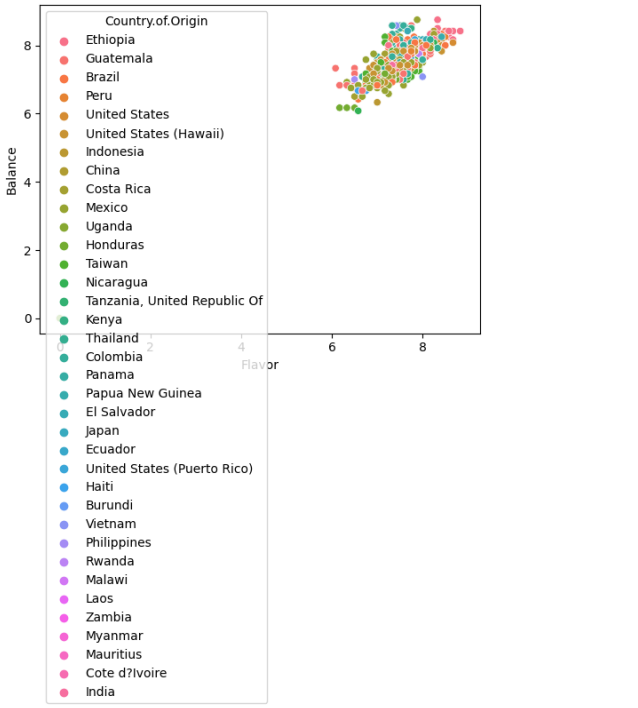
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



From this we can see that the color doesn't appear to be related to the flavor or balance scores, but that the flavor and balacne are related.

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin')
```

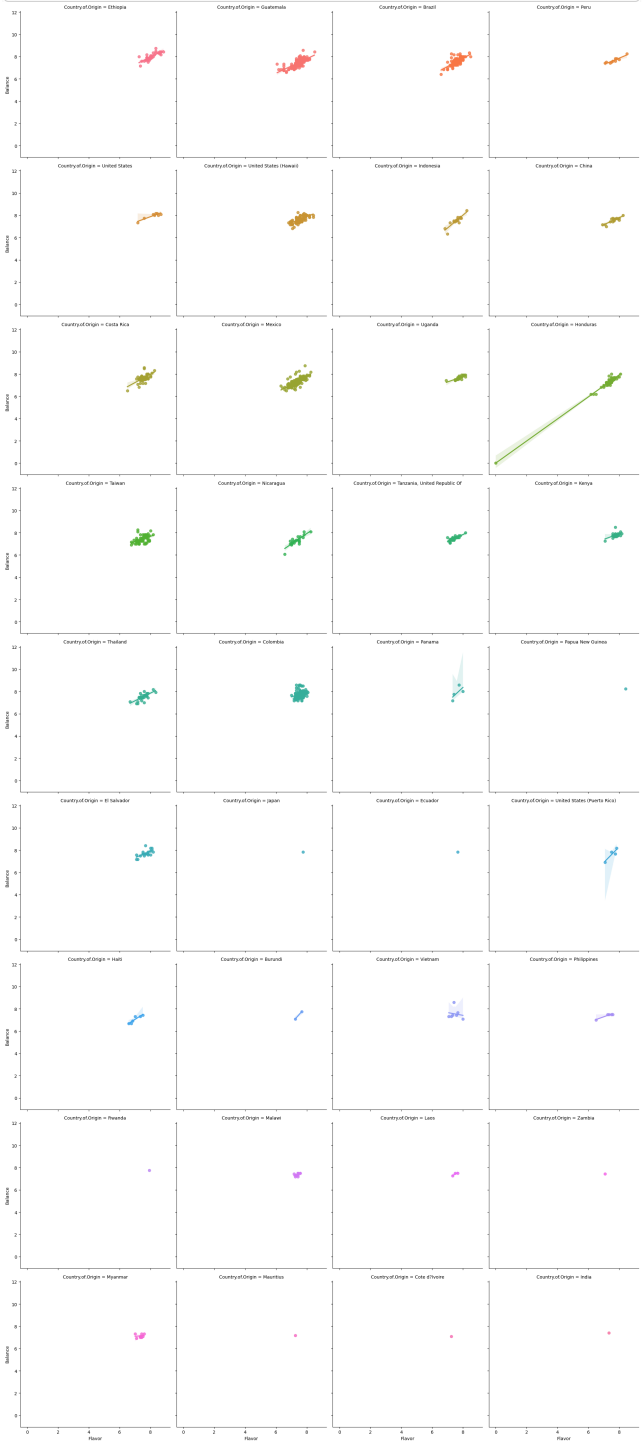
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



We can also break this apart. `lmplot` is a higher level plotting function so it allows us to create grids of plots and by default also includes a regression line.

```
sns.lmplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin',col='Country.of.Origin',
           col_wrap=4)
```

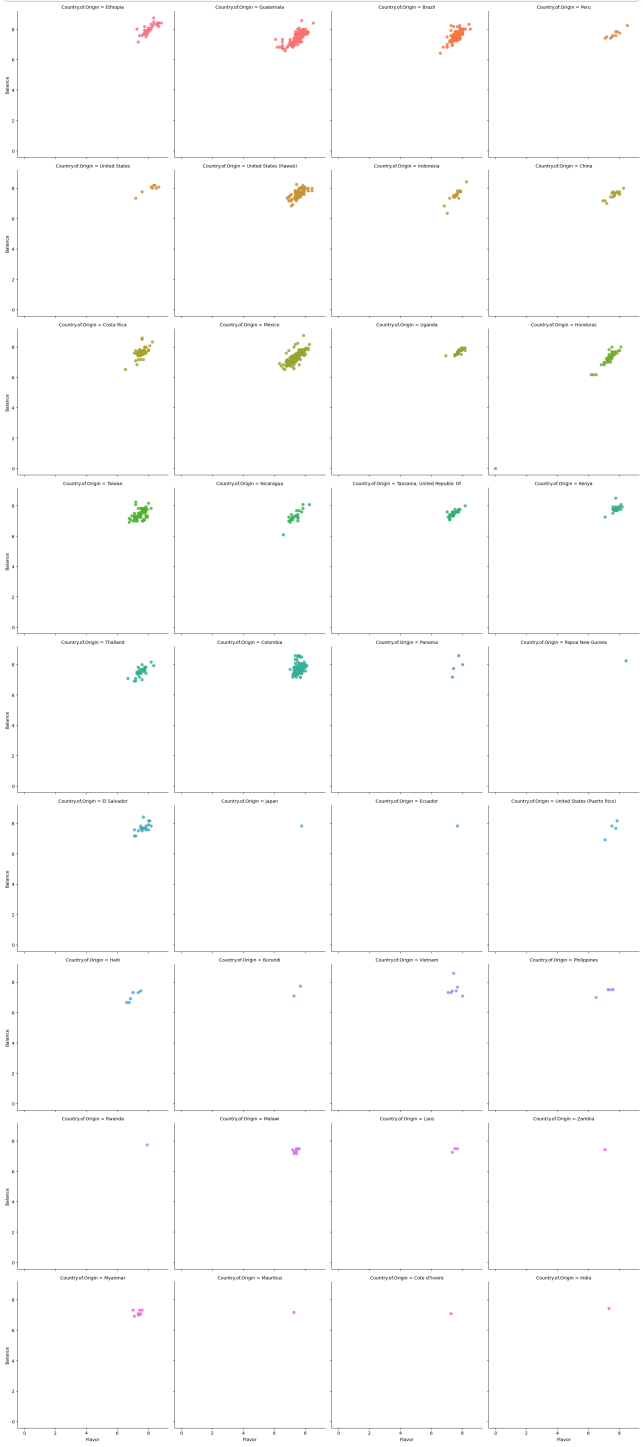
```
<seaborn.axisgrid.FacetGrid at 0x7fad6f41b990>
```



If we were not interested in the regression line, we could turn that off for now, with `,fit_reg=False`.

```
sns.lmplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin',col='Country.of.Origin',
           col_wrap=4,fit_reg=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fad65091f10>
```



7.5. Things you want to know more about

7.5.1. Things we will touch Friday

- more parameters and capabilities for seaborn
- how to analyze a graph

7.5.2. Things that you could study for level 3

- The little details about the plot functions
- More things that you can do with seaborn, like how to manipulate the scatterplots more
- More ways to create unique plots and charts (we'll give you a bit more depth but all the way here will be a level 3 task)
- One thing I want to learn based on what's learned so far is if we can compare graphs explicitly.

7.5.3. Things we will do later

7.5.3.1. Plotting data from multiple datasets on the same graphs (considering they share the same data points)

This will require being able to combine datasets into a single DataFrame, because a single plot funciton will require

7.5.4. Graphing lines of best fit onto graphs (but not just linear models, quadratic or other types as well)

We will learn how to fit models in general when we get to regression, but ultimately quadratic and other polynomial plots will also be a level 3 exploration you can do. We will get you to the point in class of all the pieces, but you will have to swap in some alternative parametrs

7.5.5. How to remove outliers or other data.

When we saw indexing we got really close to this, and we will do more when we clean data next week.

8. More EDA

```
import pandas as pd
import seaborn as sns
sns.set_theme(palette='colorblind')
```

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
robusta_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

```
coffee_df = pd.read_csv(arabica_data_url,index_col=0)
```

8.1. Which of the following is distributed most similarly to Sweetness?

```
scores_of_interest = ['Flavor','Balance','Aroma','Body',
                      'Uniformity','Aftertaste','Sweetness']
```

We can answer this statistically, technically, but it will be easiest to do this looking ad a plot.

First step is to subset the data:

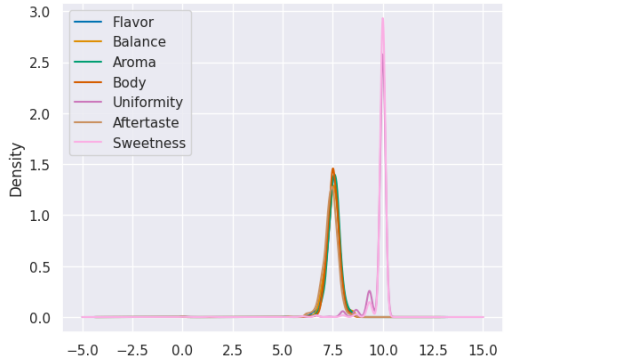
```
coffee_df[scores_of_interest].head(2)
```

	Flavor	Balance	Aroma	Body	Uniformity	Aftertaste	Sweetness
1	8.83	8.42	8.67	8.50	10.0	8.67	10.0
2	8.67	8.42	8.75	8.42	10.0	8.50	10.0

Then we produce a kde plot

```
coffee_df[scores_of_interest].plot(kind='kde')
```

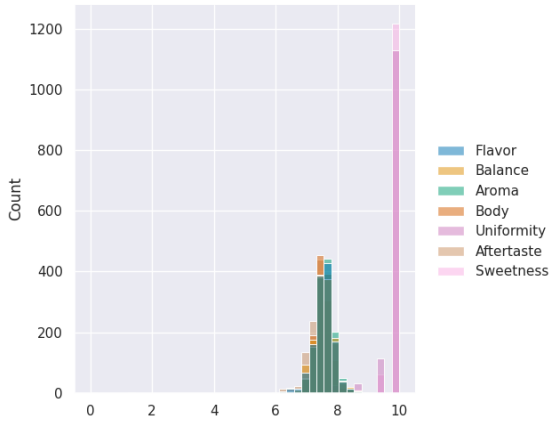
```
<AxesSubplot:ylabel='Density'>
```



We could also do it with seaborn

```
sns.displot(data=coffee_df[scores_of_interest])
```

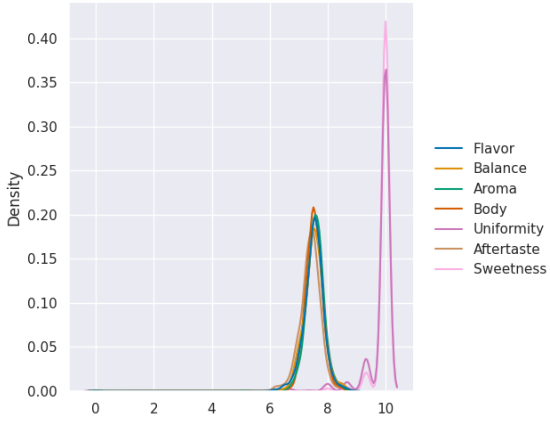
```
<seaborn.axisgrid.FacetGrid at 0x7f90e523a290>
```



the default is not as helpful as using

```
sns.displot(data=coffee_df[scores_of_interest],kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e4e7c650>
```



If we forget the parameter `kind`, we get its default value.

```
print('\n'.join(sns.displot.__doc__.split('\n')[5:10]))
```

```
``kind`` parameter selects the approach to use:

- :func:`histplot` (with ``kind="hist"``; the default)
- :func:`kdeplot` (with ``kind="kde"``)
- :func:`ecdfplot` (with ``kind="ecdf"``; univariate-only)
```

Note

If you show this excerpt, you'll see how I was able to select only a subset of the docstring to display in the notebook, programmatically. You're not required to know how to do it, but if you're curious, you can see.

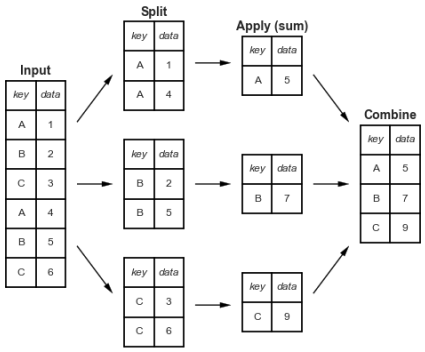
8.2. Summarizing with multiple variables

So, we can summarize data now, but the summaries we have done so far have treated each variable one at a time. The most interesting patterns are in often in how multiple variables interact. We'll do some modeling that looks at multivariate functions of data in a few weeks, but for now, we do a little more with summary statistics.

On Monday, we saw how to see how many reviews there were per country, using

```
total_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

What just happened?



Groupby splits the whole dataframe into parts where each part has the same value for `Country.of.Origin` and then after that, we extracted the `Number.of.Bags` column, took the sum (within each separate group) and then put it all back together in one table (in this case, a `Series` becuase we picked one variable out)

8.2.1. How does Groupby Work?

Important

This is more details with code examples on how the groupby works. If you want to run this code for yourself, use the download icon at the top right to download these notes as a notebook.

We can view this by saving the groupby object as a variable and exploring it.

```
country_grouped = coffee_df.groupby('Country.of.Origin')

country_grouped
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f90e4d30a50>
```

Trying to look at it without applying additional functions, just tells us the type. But, it's iterable, so we can loop over.

```
for country,df in country_grouped:
    print(type(country), type(df))
```

```
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
```

We could manually compute things using the data structure, if needed, though using pandas functionality will usually do what we want. For example:

```python
bag_total_dict = {}

for country,df in country_grouped:
    tot_bags =  df['Number.of.Bags'].sum()
    bag_total_dict[country] = tot_bags

pd.DataFrame.from_dict(bag_total_dict, orient='index',
                           columns = ['Number.of.Bags.Sum'])
```

	Number.of.Bags.Sum
Brazil	30534
Burundi	520
China	55
Colombia	41204
Costa Rica	10354
Cote d?Ivoire	2
Ecuador	1
El Salvador	4449
Ethiopia	11761
Guatemala	36868
Haiti	390
Honduras	13167
India	20
Indonesia	1658
Japan	20
Kenya	3971
Laos	81
Malawi	557
Mauritius	1
Mexico	24140
Myanmar	10
Nicaragua	6406
Panama	537
Papua New Guinea	7
Peru	2336
Philippines	259
Rwanda	150
Taiwan	1914
Tanzania, United Republic Of	3760
Thailand	1310
Uganda	3868
United States	361
United States (Hawaii)	833
United States (Puerto Rico)	71
Vietnam	10
Zambia	13

is the same as what we did before

> **Note**
>
> I used this feature to build the separate view of the communication channels on this website. You can view that source using the github icon on that page.

> **Note**
>
> I tried putting this dictionary into the dataframe for display purposes using the regular constructor and got an error, so I googled about making one from a dictionary to get the docs, which is how I learned about the `from_dict` method and its `orient` parameter which solved my problems.

8.3. Sorting DataFrames

We saved the totals, so we can check what type that is.

```
type(total_per_country)
```

```
pandas.core.series.Series
```

It's a pandas series, so we can use the `sort_values` method.

```
total_per_country.sort_values(ascending=False)
```

```
Country.of.Origin
Colombia                        41204
Guatemala                       36868
Brazil                          30534
Mexico                          24140
Honduras                        13167
Ethiopia                        11761
Costa Rica                      10354
Nicaragua                        6406
El Salvador                      4449
Kenya                            3971
Uganda                           3868
Tanzania, United Republic Of     3760
Peru                             2336
Taiwan                           1914
Indonesia                        1658
Thailand                         1310
United States (Hawaii)            833
Malawi                            557
Panama                            537
Burundi                           520
Haiti                             390
United States                     361
Philippines                       259
Rwanda                            150
Laos                               81
United States (Puerto Rico)        71
China                              55
India                              20
Japan                              20
Zambia                             13
Myanmar                            10
Vietnam                            10
Papua New Guinea                    7
Cote d?Ivoire                       2
Ecuador                             1
Mauritius                           1
Name: Number.of.Bags, dtype: int64
```

Try it yourself

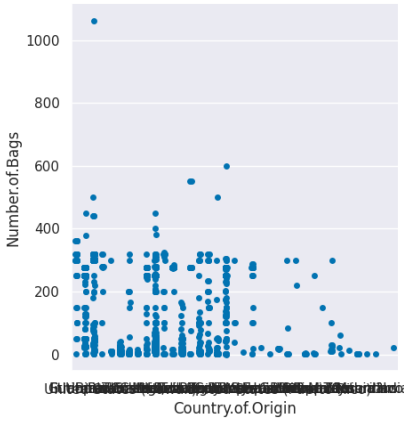
There is another sort method, `sort_index` what would that one do?

8.4. More plot types

We can also look at the distributions

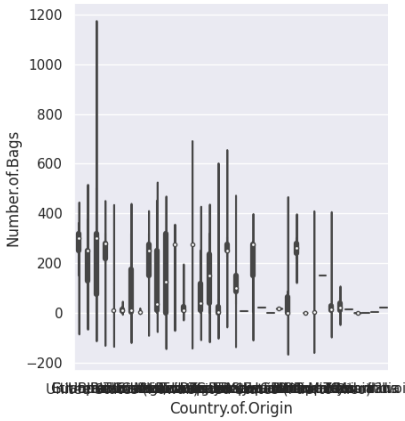
```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e4ce77d0>
```



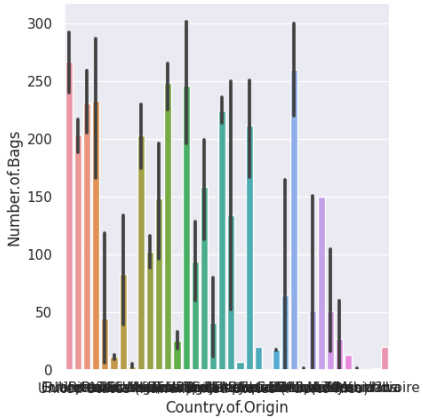

```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='violin')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e523a850>
```



```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e4b62d10>
```



8.5. How can we pick the top 10 countries out?

> **Note**
>
> We can use `head(10)` here too.

```
total_per_country.sort_values(ascending=False)[:10]
```

```
Country.of.Origin
Colombia       41204
Guatemala      36868
Brazil         30534
Mexico         24140
Honduras       13167
Ethiopia       11761
Costa Rica     10354
Nicaragua       6406
El Salvador     4449
Kenya           3971
Name: Number.of.Bags, dtype: int64
```

```
type(total_per_country.sort_values(ascending=False)[:10])
```

```
pandas.core.series.Series
```

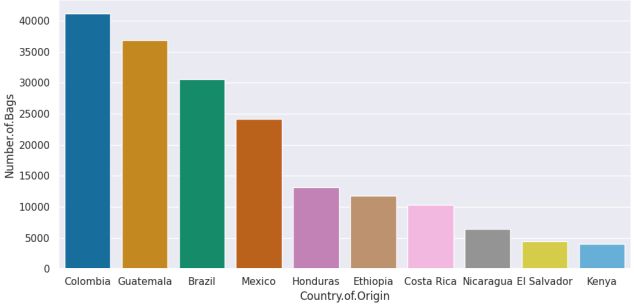
this is a series, but we can change it to a DataFrame with `reset_index()` this will also add a new index column but a side effect is that it becomes a DataFrame again

```
top_total_df = total_per_country.sort_values(ascending=False)[:10].reset_index()
top_total_df
```

	Country.of.Origin	Number.of.Bags
0	Colombia	41204
1	Guatemala	36868
2	Brazil	30534
3	Mexico	24140
4	Honduras	13167
5	Ethiopia	11761
6	Costa Rica	10354
7	Nicaragua	6406
8	El Salvador	4449
9	Kenya	3971

```
sns.catplot(data=top_total_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e4af3b50>
```



this got the number of countries fewer, bu they were still hard to read because they were small and still overlapping, so I added `aspect=2` to make it 2x as wide as tall and give it more space. We can also use a similar strategy to pull out just these country names and get all the data for those.

```
top_countries = total_per_country.sort_values(ascending=False)[:10].index
```

```
coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)].head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expiration	Certifica
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	guji-hambela	...	Green	0	April 3rd, 2016	Develo
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	guji-hambela	...	Green	1	April 3rd, 2016	Develo

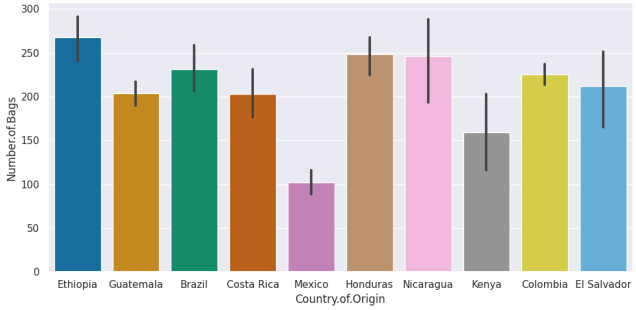
2 rows × 43 columns

I forgot the `isin` method in class and had to look that up, I more often need to filter by numerical columns or selecting a single value.

```
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)]
```

```
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e44a6ed0>
```

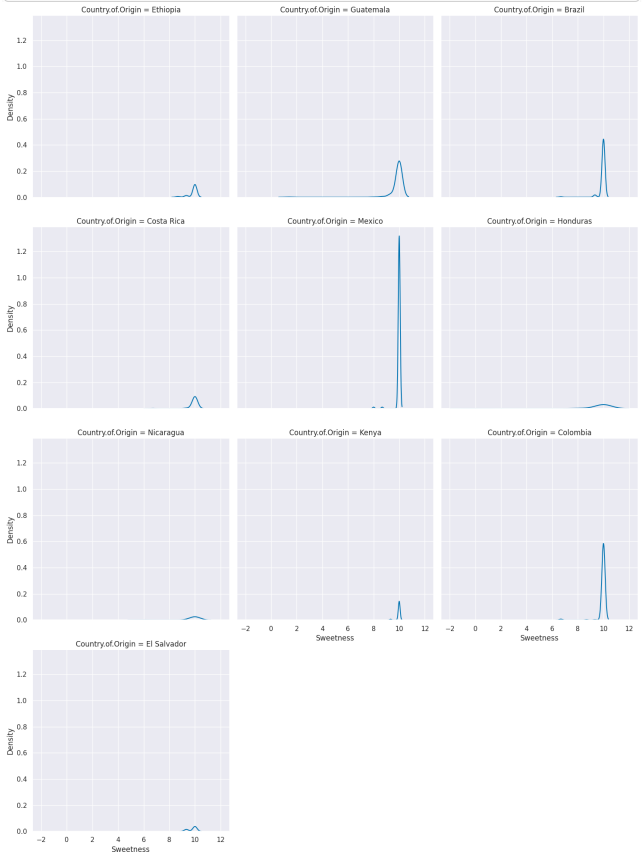



```
top_coffee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 952 entries, 1 to 1312
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Species                952 non-null    object
 1   Owner                  945 non-null    object
 2   Country.of.Origin      952 non-null    object
 3   Farm.Name              694 non-null    object
 4   Lot.Number             205 non-null    object
 5   Mill                   760 non-null    object
 6   ICO.Number             908 non-null    object
 7   Company                789 non-null    object
 8   Altitude               833 non-null    object
 9   Region                 919 non-null    object
 10  Producer               812 non-null    object
 11  Number.of.Bags         952 non-null    int64
 12  Bag.Weight             952 non-null    object
 13  In.Country.Partner     952 non-null    object
 14  Harvest.Year           932 non-null    object
 15  Grading.Date           952 non-null    object
 16  Owner.1                945 non-null    object
 17  Variety                824 non-null    object
 18  Processing.Method      850 non-null    object
 19  Aroma                  952 non-null    float64
 20  Flavor                 952 non-null    float64
 21  Aftertaste             952 non-null    float64
 22  Acidity                952 non-null    float64
 23  Body                   952 non-null    float64
 24  Balance                952 non-null    float64
 25  Uniformity             952 non-null    float64
 26  Clean.Cup              952 non-null    float64
 27  Sweetness              952 non-null    float64
 28  Cupper.Points          952 non-null    float64
 29  Total.Cup.Points       952 non-null    float64
 30  Moisture               952 non-null    float64
 31  Category.One.Defects   952 non-null    int64
 32  Quakers                951 non-null    float64
 33  Color                  808 non-null    object
 34  Category.Two.Defects   952 non-null    int64
 35  Expiration             952 non-null    object
 36  Certification.Body     952 non-null    object
 37  Certification.Address  952 non-null    object
 38  Certification.Contact  952 non-null    object
 39  unit_of_measurement    952 non-null    object
 40  altitude_low_meters    830 non-null    float64
 41  altitude_high_meters   830 non-null    float64
 42  altitude_mean_meters   830 non-null    float64
dtypes: float64(16), int64(3), object(24)
memory usage: 359.5+ KB
```

```
sns.displot(top_coffee_df, x='Sweetness', kind='kde',
col='Country.of.Origin',col_wrap=3)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e4675dd0>
```



We'll see next week how to manipulate the dataset so that we can plot multiple scores this way.

Variable types and data types Related but not the same.

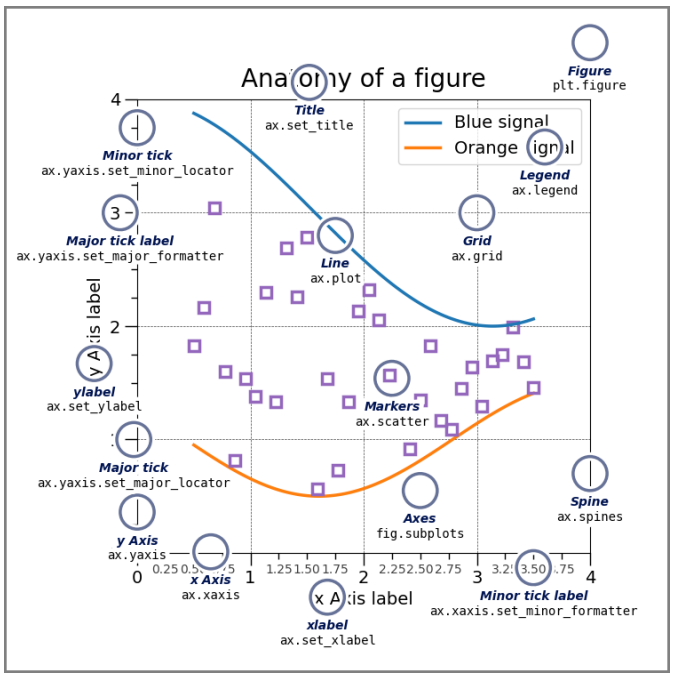
8.6. General Plotting Ideas

There are lots of type of plots, we saw the basic patterns of how to use them and we've used a few types, but we cannot (and do not need to) go through every single type. There are general patterns that you can use that will help you think about what type of plot you might want and help you understand them to be able to customize plots.

[Seaborn's main goal is opinionated defaults and flexible customization](https://seaborn.pydata.org/tutorial/introduction.html#opinionated-defaults-and-flexible-customization

8.6.1. Anatomy of a figure

First is the matplotlib structure of a figure. BOth pandas and seaborn and other plotting libraries use matplotlib. Matplotlib was used in visualizing the first Black hole.



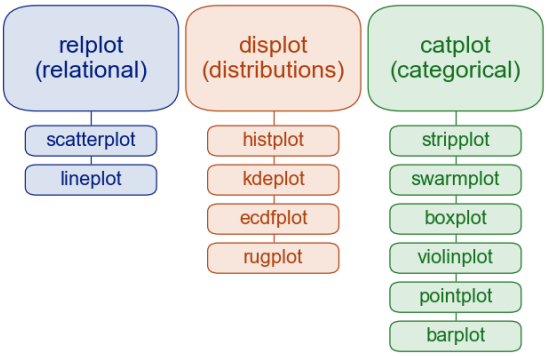
This is a lot of information, but these are good to know things. THe most important is the figure and the axes.

> **Try it Yourself**
>
> Make sure you can explain what is a figure and what are axes in your own words and why that distinction matters. Discuss in office hours if you are unsure.

that image was drawn with code and that page explains more.

8.6.2. Plotting Function types in Seaborn

Seaborn has two *levels* or groups of plotting functions. Figure and axes. Figure level fucntions can plot with subplots.



This is from thie overivew section of the official seaborn tutorial. It also includes a comparison of figure vs axes plotting.

The official introduction is also a good read.

8.6.3. More

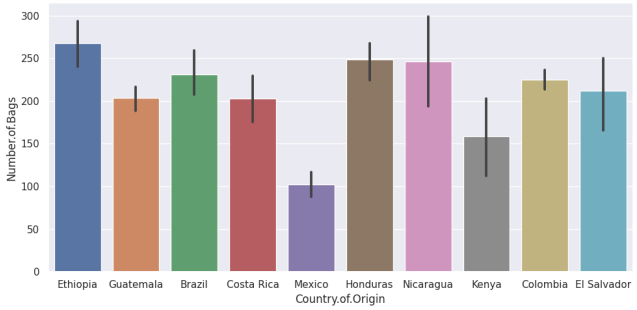
The seaborn gallery and matplotlib gallery are nice to look at too.

8.7. Styling Plots

```
sns.set_theme()
```

```
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

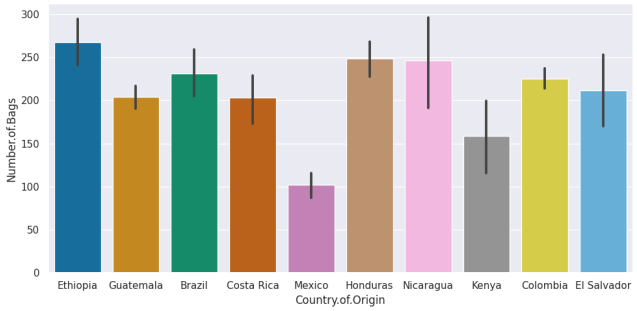
```
<seaborn.axisgrid.FacetGrid at 0x7f90e3385cd0>
```



This by default styles the plots to be more visually appealing

```
sns.set_theme(palette='colorblind')
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

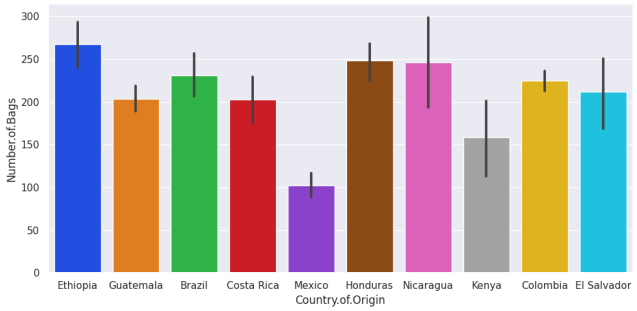
```
<seaborn.axisgrid.FacetGrid at 0x7f90e331b190>
```



the colorblind palette is more distinguishable under a variety fo colorblindness types. for more

```
sns.set_theme(palette='bright')
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f90e33bb250>
```

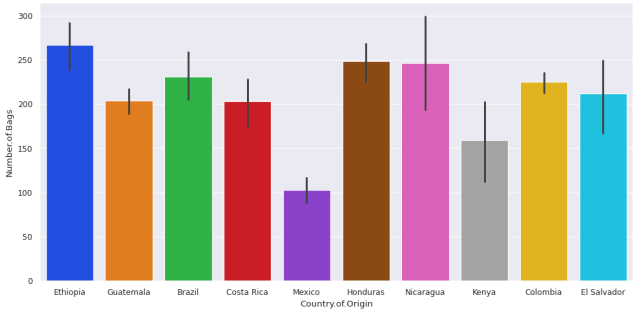


Colorblind is a good default, but you can choose others that yo like more too.

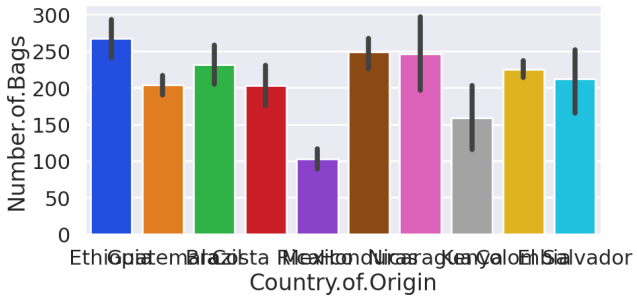
[more on colors]

to prepare plots for posters vs printing, etc you can use:

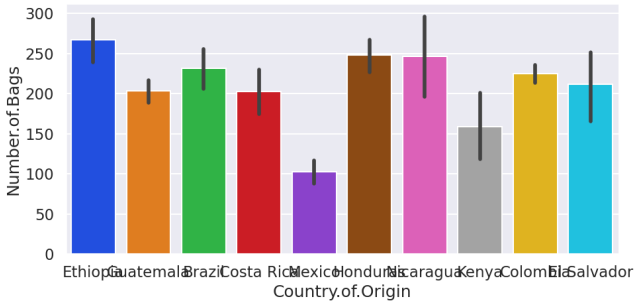
```
with sns.plotting_context('paper'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```



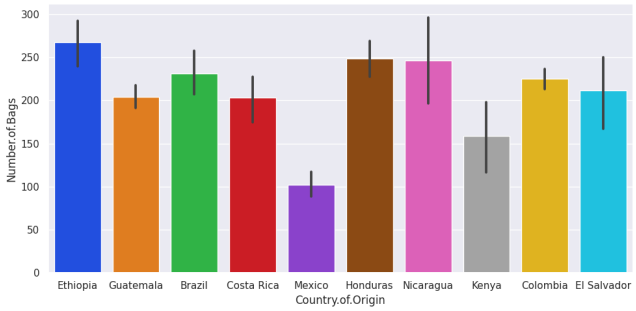

```
with sns.plotting_context('poster'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```




```
with sns.plotting_context('talk'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```




```
with sns.plotting_context('notebook'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```



It's not perfect, but it gives you a good starting point.

8.8. More Practice

- Make a table thats total number of bags and mean and count of scored for each of the variables in the `scores_of_interest` list.
- Make a bar chart of the mean score for each variable `scores_of_interest` grouped by country.

8.9. Questions

> **Note**
>
> Submit questions as Issues

8.10. Questions After Class

9. Intro to Data Cleaning

This week, we'll be cleaning data.

Cleaning data is **labor intensive** and requires making *subjective* choices.
We'll focus on, and assess you on, manipulating data correctly, making reasonable choices, and documenting the choices you make carefully.

We'll focus on the programming tools that get used in cleaning data in class this week:

- reshaping data
- handling missing or incorrect values
- renaming columns

```
import pandas as pd
import seaborn as sns

# sns.set_theme(pallete='colorblind')
```

9.1. Tidy Data

Read in the three csv files described below and store them in a list of dataFrames

```
url_base = 'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'

datasets = ['study_a.csv','study_b.csv','study_c.csv']
```

```
df_list = [pd.read_csv(url_base + current) for current in datasets]
```

```
type(df_list)
```

```
list
```

```
type(df_list[0])
```

```
pandas.core.frame.DataFrame
```

```
df_list[0]
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

```
df_list[1]
```

	intervention	John Smith	Jane Doe	Mary Johnson
0	treatmenta	-	16	3
1	treatmentb	2	11	1

```
df_list[2]
```

	person	treatment	result
0	John Smith	a	-
1	Jane Doe	a	16
2	Mary Johnson	a	3
3	John Smith	b	2
4	Jane Doe	b	11
5	Mary Johnson	b	1

These three all show the same data, but let's say we have two goals:

- find the average effect per person across treatments
- find the average effect per treatment across people

This works differently for these three versions.

```
df_list[0].mean()
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in
DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version
this will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

```
treatmenta   -54.333333
treatmentb     4.666667
dtype: float64
```

we get the average per treatment, but to get the average per person, we have to go across rows, which we can do here, but doesn't work as well with plotting

```
df_list[0].mean(axis=1)
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in
DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version
this will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

```
0     2.0
1    11.0
2     1.0
dtype: float64
```

and this is not well labeled.

Let's try the next one.

```
df_list[1].mean()
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in
DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version
this will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

```
John Smith      -1.0
Jane Doe        13.5
Mary Johnson     2.0
dtype: float64
```

Now we get the average per person, but what about per treatment? again we have to go across rows instead.

```
df_list[1].mean(axis=1)
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in
DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version
this will raise TypeError.  Select only valid columns before calling the
reduction.
  """Entry point for launching an IPython kernel.
```

```
0    9.5
1    6.0
dtype: float64
```

For the third one, however, we can use groupby

```
df_list[2].groupby('person').mean()
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3553: FutureWarning: Dropping invalid
columns in DataFrameGroupBy.mean is deprecated. In a future version, a TypeError
will be raised. Before calling .mean, select only columns which should be valid
for the function.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	result
person	
Jane Doe	805.5
John Smith	-1.0
Mary Johnson	15.5

```
df_list[2].groupby('treatment').mean()
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3553: FutureWarning: Dropping invalid
columns in DataFrameGroupBy.mean is deprecated. In a future version, a TypeError
will be raised. Before calling .mean, select only columns which should be valid
for the function.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	result
treatment	
a	-54.333333
b	703.666667

The original Tidy Data paper is worth reading to build a deeper understanding of these ideas.

9.2. Tidying Data

Let's reshape the first one to match the tidy one. First, we will save it to a DataFrame, this makes things easier to read and enables us to use the built in help in jupyter, because it can't check types too many levels into a data structure.

```
treat_df = df_list[0]
```

Let's look at it again, so we can see

```
treat_df.head()
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

```
df_list[0].columns
```

```
Index(['name', 'treatmenta', 'treatmentb'], dtype='object')
```

```
df_a = df_list[0]
```

```
df_a.melt(id_vars=['name'],value_vars=['treatmenta','treatmentb'],
          value_name='result',var_name='treatment')
```

	name	treatment	result
0	John Smith	treatmenta	-
1	Jane Doe	treatmenta	16
2	Mary Johnson	treatmenta	3
3	John Smith	treatmentb	2
4	Jane Doe	treatmentb	11
5	Mary Johnson	treatmentb	1

When we melt a dataset:

- the `id_vars` stay as columns
- the data from the `value_vars` columns become the values in the `value` column
- the column names from the `value_vars` become the values in the `variable` column
- we can rename the value and the variable columns.

Let's do it for our coffee data:

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'
```

```
coffee_df = pd.read_csv(arabica_data_url)
scores_of_interest = ['Balance','Aroma','Body','Aftertaste']
```

```
coffee_tall = coffee_df.melt(id_vars=['Country.of.Origin','Color'],
                             value_vars=scores_of_interest
              ,var_name = 'Score')
coffee_tall.head()
```

	Country.of.Origin	Color	Score	value
0	Ethiopia	Green	Balance	8.42
1	Ethiopia	Green	Balance	8.42
2	Guatemala	NaN	Balance	8.42
3	Ethiopia	Green	Balance	8.25
4	Ethiopia	Green	Balance	8.33

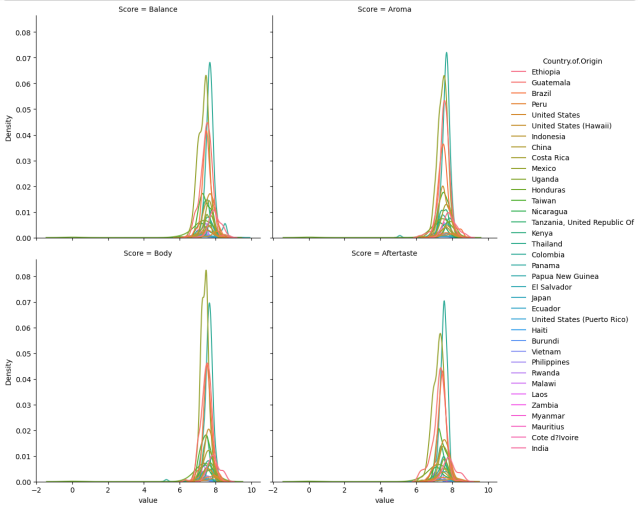
Notice that the actual column names inside the `scores_of_interest` variable become the values in the variable column.

This one we can plot in more ways:

```
sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
            col='Score',kind='kde',col_wrap=2)
```

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:2: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
```

```
<seaborn.axisgrid.FacetGrid at 0x7f1a6c9e95d0>
```



9.3. Filtering Data

```
high_prod = coffee_df[coffee_df['Number.of.Bags']>250]
high_prod.shape
```

```
(368, 44)
```

```
coffee_df.shape
```

```
(1311, 44)
```

We see that filters and reduces. We can use any boolean expression in the square brackets.

```
top_balance = coffee_df[coffee_df['Balance']>coffee_df['Balance'].quantile(.75)]
top_balance.shape
```

```
(252, 44)
```

We can confirm that we got only the top 25% of balance scores:

```
top_balance.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total
count	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	
mean	273.535714	153.337302	7.808889	7.837659	7.734167	7.824881	7.780159	8.003095	9.885278	9.896667	9.885952	7.865714	
std	284.249040	126.498576	0.355319	0.318172	0.302481	0.320253	0.317712	0.213056	0.363303	0.470514	0.380433	0.383738	
min	1.000000	0.000000	5.080000	7.170000	6.920000	7.080000	5.250000	7.830000	6.670000	5.330000	6.670000	5.170000	
25%	65.750000	12.750000	7.647500	7.670000	7.500000	7.670000	7.670000	7.830000	10.000000	10.000000	10.000000	7.670000	
50%	171.500000	165.500000	7.780000	7.830000	7.750000	7.830000	7.750000	7.920000	10.000000	10.000000	10.000000	7.830000	
75%	378.250000	275.000000	8.000000	8.000000	7.920000	8.000000	7.920000	8.080000	10.000000	10.000000	10.000000	8.080000	
max	1260.000000	360.000000	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.000000	10.000000	9.250000	

We can also use the `isin` method to filter by comparing to an iterable type

```
total_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
top_countries = total_per_country.sort_values(ascending=False)[:10].index
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)]
```

9.4. Questions after class

9.4.1. In the data we had about treatment a and treatment b. Say there was another column that provided information about the age of the people. Could we create another value variable to or would we put it in the value_vars list along with treatment a and treatmentb?

We can have multiple variables as the `id_vars` so that the dataset would have 4 columns instead of 3.

9.4.2. Can we clean data within any row of the data as long as there is a column name for it?

Yes

9.4.3. How to clean larger datasets

Everything we will learn will work in any context that you can load the dataset into RAM on the computer you are working on; or process it in batches.

9.4.4. with very large data sets, how can you tell id there are missing values or incorrect types present?

We use ways of checking the values, like the `info` method to check the whole column.

9.4.5. Are there any prebuilt methods to identify and remove extreme outliers?

There may be, that is a good thing to look in the documentation for. However, typically the definition of an outlier is best made within context, so the filtering strategy that we just used would be the way to remove them, after doign some EDA.

9.4.6. Is there a specific metric to which a dataset must meet to consider it 'cleaned'?

It's "clean" when it will work for the analysis you want to do. This means clean enough for one analysis may not be enough for another goal. Domain expertise is alwasy important.

9.4.7. Are there any packages we may utilize that help us clean data more effectively?

9.4.8. Things we will do later this week:

- adding more data to a DataFrame in jupyter notebook?
- replace values
- deal with NaN values in a dataset?

9.4.9. Next week

- What is the difference between different types of merging data frames (inner, outer, left, right, etc)?

10. Fixing Data representations

10.1. Admin

- next assignment posted tonight.

10.1.1. a4 TLDR:

- Review how your new dataset manipulation skills could have helped you in A2 or A3.
- clean provided (in the template) datasets
  - do tiny EDA to show complete;
  - add more EDA to get summarize and visualize
  - clean a specific dataset for access level 2 or python level 2 also; do both if you need all 3 achievments (prepare, access, python)
- Study some examples and notice patterns in data clearning

10.1.2. Portfolio update

- merge in your work from assignment 1
- portoflio will be due in ~2 weeks start planning
- read the instructions and example ideas; there will be time for q&a on Friday
- you can also create an outline and get feedback on your plan using an issue on your portflio repo

Note

The portfolio is open ended intentionally. I want you to learn these skills a little bit deeper than you have for the assignments, which is a little bit deeper than we cover in class and show me that you learned. I do *not* want you to spend too much effort guessing what I want. I will look at what you submit for evidence of learning and assess that. There is not a single "right" answer.

10.2. Review Filtering

```
import pandas as pd
```

We can also filter using the `isin` method to compare each item in a column to a list

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'
# load the data
coffee_df = pd.read_csv(arabica_data_url)
# get total bags per country
bags_per_country_df = coffee_df.groupby('Country.of.Origin')
['Number.of.Bags'].sum()

# sort descending, keep only the top 10 and pick out only the country names
top_bags_country_list = bags_per_country_df.sort_values(ascending=False)
[:10].index

# filter the original data for only the countries in the top list
top_coffee_df =
coffee_df[coffee_df['Country.of.Origin'].isin(top_bags_country_list)]
```

Note

You can see more about groupby in notes from last week.

Yes this is a Series, not a data Frame, but a DataFrame is built of Series, so this is a small error in naming. `top_bags_country_list` is also an `Index` not a `list` but the names need to give an idea, not necessarily be precise so my choices are okay, but could be better.

```
type(coffee_df['Number.of.Bags'])
```

```
pandas.core.series.Series
```

```
type(coffee_df.loc[1])
```

```
pandas.core.series.Series
```

We can look at the final result

```
top_coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Expiratio
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	April 3r 201
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	April 3r 201
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	May 31s 201
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	Marc 25th, 201
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	April 3r 201

5 rows × 44 columns

```
top_coffee_df.shape, coffee_df.shape
```

```
((952, 44), (1311, 44))
```

10.3. Fixing a Column

In tidy data each column is exactly one variable. Let's look at the `Bag.Weight`

```
coffee_df['Bag.Weight'].sample(10)
```

```
552     70 kg
99      30 kg
272      1 kg
996      1 kg
426     70 kg
678     69 kg
1057     1 kg
1159     1 kg
828      1 kg
951     69 kg
Name: Bag.Weight, dtype: object
```

This is actually two pieces of information, the value measured and the units used. In addition to the fact that there are multiple units, even if we could convert them, we cannot do math on these because they're strings. (notice it says "object" as the type)

Series have a `str` attribute so we can apply base python string methods to each value in a column.

```
coffee_df['Bag.Weight'].str
```

```
<pandas.core.strings.accessor.StringMethods at 0x7f29b6479190>
```

What we want is to split it

```
coffee_df['Bag.Weight'].str.split(' ').sample(10)
```

```
908     [60, kg]
1233     [1, kg]
1121     [1, kg]
404     [3, lbs]
594     [69, kg]
1010     [1, kg]
640     [20, kg]
1137     [1, kg]
1105     [2, kg]
647     [70, kg]
Name: Bag.Weight, dtype: object
```

Since this looks good, we can save it to a variable.

```
split_bw = coffee_df['Bag.Weight'].str.split(' ')
```

We still have one problem, each element contains a list.

```
type(split_bw[0])
```

```
list
```

What we want is a `DataFrame` with two columns, one of the first value and one of the second value for each list.

A `DataFrame` is build of Series which are each row (and each column) the lists we have are each the content that we want to put in one Series and then stack them all together.

To do this, we can cast each list to a `Series` using the `apply` method which automatically stacks Series or DataFrames back together after applying a function to each row (or column).

Recall that in python, we can use types as functions to cast

```
num = '2'
```

```
type(int(num))
```

```
int
```

```
type(num)
```

```
str
```

So, back to our real problem:

```
split_bw.apply(pd.Series)
```

	0	1
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg
...	...	...
1306	1	kg
1307	2	kg
1308	69	kg
1309	1	kg
1310	69	kg

1311 rows × 2 columns

This looks good, but these columns are not very informative, so we can rename them.

Rename can take many different inputs and be applied on lots of parts, but we will use it with a dictionary that serves as a mapping (like the mathematical sense of mapping; like a function). It will change the column with each key to the value.

```
split_df = split_bw.apply(pd.Series).rename(columns={0:'Weight',1:'Units'})
split_df
```

	Weight	Units
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg
...	...	...
1306	1	kg
1307	2	kg
1308	69	kg
1309	1	kg
1310	69	kg

1311 rows × 2 columns

This is good, but it's only the one column. We can use concat to put them together.

```
pd.concat([coffee_df,split_df],axis=1).head(2)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Expiration	Certification.Body	
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	April 3rd, 2016	METAD Agricultural Development plc	309fcf77415a3661
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	April 3rd, 2016	METAD Agricultural Development plc	309fcf77415a3661

2 rows × 46 columns

Why `axis=1` again?

Let's look at the other one.

```
bad_concat = pd.concat([coffee_df,split_df],axis=0)
bad_concat.shape
```

```
(2622, 46)
```

It has double the rows

> **Important**
>
> checking the shame is the first thing to do to see if you did the right one

and the bottom of it most columns are NaN (null, missing)

```
bad_concat.tail()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Expiration	Certification.Body	Certification.Add
1306	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1308	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1310	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

5 rows × 46 columns

1. Portfolio Setup, Data Science, and Python

Due: 2020-09-11

1.1. Objective & Evaluation

This assignment is an opportunity to earn level 1 achievements for the `process` and `python` and confirm that you have all of your tools setup, including your portfolio.

> **Note**
>
> Typically, assignments are for level 2, but this week we did setup stuff in class instead of a comprehension checks. Also, the data science process takes a bit more time to really familiar with, it's hard to explain the modeling step for example, without knowing how to build models.
>
> You will be able to earn python level 2 starting in assignment 2 and process level 2 starting with assignment 6.

1.2. To Do

Your task is to:

1. Install required software from the Tools & Resource page
2. Create your portfolio, by accepting the assignment
3. Learn about your portfolio from the README file on your repository.
4. edit `_config.yml` to set your name as author and (optionally) change the logo image
5. Fill in `about/index.md` with information about yourself(not evaluated, but useful) and your own definition of data science (graded for **level 1 process**)
6. Add a Jupyter notebook called `grading.ipynb` to the `about` folder and write a function that computes a grade for this course, with the following docstring. Include:
   - 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.
   - a basic function that uses conditionals in python will earn **level 1 python**
7. Add the line `- file: about/grading` in your `_toc.yml` file.

> **Important**
>
> remember to add, commit, and push your changes so we can see them

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

    Parameters:
    ------------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

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

> **Note**
>
> If you get stuck on any of this after accepting the assignment and creating a repository, you can create an issue on your repository, describing what you're stuck on and tag us: @rhodypro4dg/fall22instructors
>
> To do this click Issues at the top, the green "New Issue" button and then type away.

> **Important**
>
> If you have trouble, check the GitHub FAQ on the left before e-mailing

1.3. Tips

Here are some sample tests you could run to confirm that your function works correctly:

```
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+'
```

> **Warning**
>
> your function can have a different name than `compute_grade`, but make sure it's your function name, with those parameter values in your tests.

> **Note**
>
> when the value of the expression after `assert` is `True`, it will look like nothing happened. `assert` is used for testing

1.4. Submission Instructions

1.4.1. If using GitHub in the browser only

Create a Jupyter Notebook with your function in a portfolio folder on your computer where you will save all of your work for the portfolio. Then upload it to GitHub, following GitHub instructions for adding a file

1.4.2. If using git offline

Create a Jupyter Notebook with your function in your portfolio folder, then add, commit, and push the changes.

In your browser, view the `gh-pages` branch to see your compiled submission, as `portfolio.pdf` or by viewing your website.

1.5. Thinking Ahead

> **Note**
>
> Thinking Ahead is an optional section you can do to get a head start on the next level achievements

Add a markdown file to your portfolio called `ideas.md` and start answering some of the following questions:

1. Given what you know about the Data Science Process, which steps do you think you will like most? least?
2. What steps will use the most domain knowledge?
3. What applications of data science do you have domain knowledge for?

2. Assignment 2: Practicing Python and Accessing Data

due : 2022-09-21

2.1. Setting

Next week, we are going to learn about summarizing data. In this assignment, you are going to build a small dataset about datasets. In class next week, we will combine all of your datasets about datasets together in order to be able to answer questions like:

- how much total data did you all load
- how many students picked the same dataset?
- how many total rows of data did each student load?

2.2. Objective & Evaluation

This assignment is an opportunity to earn level 1 and 2 achievements in `python` and `access` and begin working toward level 1 for `summarize`. You can also earn level 1 for `process`.

In this assignment, you'll practice/ review python skills by manipulating datasets and extracting basic information about them.

Task	Skills (max level)
identify possible uses for data in a data science pipeline	[process (1)]
load data from one file format	[access (1)]
load data from at least two of (.csv, .tsv, .dat, database, .json)	[access (2)]
compare the data formats	[access (2)]
complete the assignment in python	[python (1)]
use python data types (eg dictionaries) to prepare information about datasets	[python (2)]
use informative variable names, pythonic iteration, and other common PEP 8 conventions	[python (2)]
display DataFrame properties	[summarize (1)]

Table 2.1 practice python by manipulating data files, load datasets of different types

First, accept the assignment . It contains a notebook with some template structure (and will set you up for grading).

2.3. Find Datasets

Find 3 datasets of interest to you that are provided in at least two different file formats. Choose datasets that are not too big, so that they do not take more than a few second to load. At least one dataset, must have non numerical (eg string or boolean) data in at least 1 column.

In your notebook, create a markdown cell for each dataset that includes:

- heading of the dataset's name
- a 1-2 sentence summary of what the dataset contains and why it was collected
- a "more info" link to where someone can learn about the dataset
- 1-2 questions you would like to answer with that dataset.

> **Important**
>
> After finding datasets, how to do the rest of these steps can be found within the course site (notes and glossary) or the pandas documentation.
>
> Learning to use the documentation effectively is important; libraries will change over time and random pages on the internet will not be updated accordingly, but in a well maintained library the documentation will get upated with changes.

2.4. Store them for loading

Create a list of dictionaries in `datasets.py`, so that there is one dictionary for each dataset. Each dictionary should have the following keys:

`url`	with the url
`short_name`	a short name
`load_function`	(the actual function handle) what function should be used to load the data into a `pandas.DataFrame`.

Table 2.2 Meta Data Description of the dictionary to create

2.5. Make a dataset about your datasets

Import the list from the `datasets` module you created in the step above. Then iterate over the list of dictionaries, and:

1. load each dataset from the url
2. save the dataset to a local csv using the short name you provided for the dataset as the file name, without writing the index column to the file.
3. record attributes about the dataset as in the table below in a list of lists or dictionary
4. Use that to create a DataFrame with columns that match the rows of the following table.

> **Note**
>
> to earn **level 2 python** use pythonic code to write a loop that tests your function's correctness, by iterating over a list or dictionary. You will have many chances to earn level 2 achievement in python so this step is optional.

> **Note**
>
> The term iterate is defined in the site glossary.

name	a short name for the dataset
source	a url to where you found the data
num_rows	number of rows in the dataset
num_columns	number of columns in the dataset
num_numerical	number of numerical variables in the dataset

Table 2.3 Meta Data Description of the DataFrame to build

> **Hint**
>
> DataFrame objects have many input/output methods beyond the read methods that we have seen so far, including methods to save a DataFrame to your computer's hard drive. Saving it to your computer makes a local (not on the internet) copy.

2.6. Explore Your Datasets

> **Hint**
>
> Notice that I refer to loading the datasets in two different ways, once from a URL, once from a relative path. What does that mean about the way that you can store it for use while you iterate?

For one dataset that includes nonnumerical data:

- load it in from your local csv using a relative path
- display the heading and the last 4 rows
- make a numpy array of only the numerical data and save it to a new variable (select these programmatically)
- was the format that the data was provided in a good format? why or why not?

For any other dataset:

- load it in from your local csv using a relative path
- display the heading with the first three rows
- display the datatype for each column
- Are there any variables where pandas may have read in the data as a datatype that's not what you expect (eg a numerical column mistaken for strings)? If so, investigate and try to figure out why.

For the third dataset:

- load it in from your local csv using a relative path
- display the first 3 multiples of 3 rows (eg 3,6,9) of the data for two columns of your choice

2.7. Exploring data files

There are two files in the data folder, both can be read in with `read_csv` but need some options or fixing.

- try to read in the `german.data` file, what happens with the default settings? What option do you need to use to make it look right?
- try to read in the `.csv` file that's included in the template repository (), use the error messages you get to try to fix the file manually (any text editor, including jupyter can edit a `.csv`), making notes about what changes you made in a markdown cell.

2.8. Submission

This time you have to separately submit from posting your code to make grading easier. Go to the actions tab and run the action called "Submit".

2.9. Thinking ahead

> **Important**
>
> his section is not required, but is intended to help you get started thinking about ideas for your portfolio. If you complete it, we'll give your feedback to help shape your ideas to get to level 3 achievements. If you want to focus only on level 2 at this moment in time, feel free to skip this part. You could also think about this after submitting the assignment, since you do not have to get a grade for it. If you want, you could discuss these ideas in office hours.

1. When might you prefer one datatype over another?
2. How does PEP 8 standard code help you be collaborative?
3. Learn about Datasheets for Datasets and find some examples, (eg this google scholar result) How could something like this impact your work as a datascientist?

3. Assignment 3: Exploratory Data Analysis

__Due:2022-09-28 11:59pm __

Template repo for submission

3.1. Objective & Evaluation

This week your goal is to do a small exploratory data analysis for two datasets of your choice.

Eligible Skills:

- summarize
- visualize
- access

3.2. Choose Datasets

Each Dataset must have at least three variables, but can have more. Both datasets must have multiple types of variables. These can be datasets you used last week, if they meet the criteria below.

3.2.1. Dataset 1 (d1)

must include at least:

- two continuous valued variables and
- one categorical variable.

3.2.2. Dataset 2 (d2)

must include at least:

- two categorical variables and
- one continuous valued variable

3.3. EDA

Use a separate notebook for each dataset, name them `dataset_01.ipynb` and `dataset_02.ipynb`.

For **each** dataset, in a dedicated notebook, complete the following:

1. Load the data to a notebook as a `DataFrame` from url or local path, if local, include the data in your repository.
2. Explore the dataset in a notebook enough to describe its structure use the heading `## Description`
    - shape
    - columns
    - variable types
    - overall summary statisics
3. Write a short description of what the data contains and what it could be used for
4. Ask and answer 4 questions by using and interpreting statistics and visualizations as appropriate. Include a heading for each question using a markdown cell and H2:`##`. Make sure your analyses meet the criteria in the check lists below.
5. Describe what, if anything might need to be done to clean or prepare this data for further analysis in a finale `## Future analysis` markdown cell in your notebook.

3.3.1. Question checklist

be sure that every question (all eight, 4 per dataset) has:

- a heading
- at least 1 statistic or plot
- interpretation that answers the question

3.3.2. Dataset 1 Checklist

make sure that your `dataset_01.ipynb` has:

- Overall summary statistics grouped by a categorical variable
- A single statistic grouped by a categorical variable
- at least one plot that uses 3 total variables
- a plot and summary table that convey the same information. This can be one statistic or many.

3.3.3. Dataset 2 Checklist

- two individual summary statistics for one variable
- one summary statistic grouped by two categorical variables
- a figure with a grid of subplots that correspond to two categorical variables
- a plot and summary table that convey the same information. This can be one statistic or many.

> **Tip**
>
> Be sure to start early and use help hours to make sure you have a plan for all of these.

3.4. Peer Review

> **Note**
>
> This is optional, but if you do a review, you only need to do one analysis each.

> **Warning**
>
> Be familiar with the collaboration policy before you choose to go this route

> **Important**
>
> get group permissions in advance

With a partner (or group of 3 where person 1 review's 2 work, 2 reviews 3, and 3 reviews 1) read your partner's notebook and complete a peer review on their pull request. You can do peer review when you have done most of your analysis, and explanation, even if some parts of the code do not work. After you each do your reviews, update your own analysis.

3.4.1. Review

In your review:

- Use inline comments to denote places that are confusing or if you see solutions to problems your classmate could not solve
- keep the questions below in mind
- Use the template below for your summary review

3.4.1.1. Review Questions

1. How was the analysis overall to read? easy? hard? cohesive? jumpy?
2. Did the data summaries tell you enough about the data to understand the analysis and anticipate what kinds of questions could be answered? If not, what questions do you still have about the data?
3. Do the questions make sense based on the data? Are they interesting questions? What could improve the questions
4. Are the statistics and plots appropriate for the questions?
5. Are the interpretations complete, clear, and consistent with the statistics and plots?
6. What could be done to make the explanations more clear and complete?
7. What additional analysis might make the analysis more compelling and clear?

3.4.1.2. Review Template

```
<!-- delete sections that are not needed -->
## Overall

This analysis was ...

## Data Summaries

- [ ] complete

To understand this analysis I still need to know ...

## Checklist

- [ ] questions fit the data
- [ ] questions are in natural language
- [ ] chosen statistics and plots match questions
- [ ] all statistics and plots have an interpretation in English

## Areas of improvement
```

3.4.2. Response

Respond to your review either inline comments, replies, and by updating your analysis accordingly.

> **Think Ahead**
>
> 1. How could you make more customized summary tables?
> 2. Could you use any of the variables in this dataset to add more variables that would make interesting ways to apply split-apply-combine? (eg thresholding a continuous value to make a categorical value)

> **Warning**
>
> This section is not required, but is intended to help you get started thinking about ideas for your portfolio. If you complete it, we'll give your feedback to help shape your ideas to get to level 3 achievements. If you want to focus only on level 2 at this moment in time, feel free to skip this part.

Portfolio

```
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:26: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
/opt/hostedtoolcache/Python/3.7.14/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:31: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
```

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

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

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

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

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

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

Upcoming Checks

Portfolio 1

More information to follow

Formatting Tips

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

Your portfolio is a jupyter book. This means a few things:

- it uses myst markdown
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

Managing Files and version

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

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

To manage with a precommit hook jupytext conversion

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

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

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

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

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

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

To manage with a gh action jupytext conversion

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

```
name: jupytext

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

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

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

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

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

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

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

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

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

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

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

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

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

Organization

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

Data Files

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

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

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

Structure of plain markdown

Use a heading like this:

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

in the file and it will appear in the sidebar.

You can also make text *italic* or **bold** with either `*asterics*` or `__underscores__` with `_one for italic_` or `**two for bold**` in either case

File Naming

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

Syncing markdown and ipynb files

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

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

Adding annotations with formatting or margin notes

You can either install jupytext 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 jupytext installed, or locally if you have installed jupytext or jupyterbook.

In your .md file use backticks to mark special content blocks

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

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

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

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

For a complete list of options, see the **`sphinx-book-theme`** documentation.

Links

Markdown syntax for links

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

Configurations

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

Show errors and continue

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

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

Using additional packages

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

FAQ

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

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

Syllabus and Grading FAQ

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

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

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

Can I submit this assignment late if …?

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

I don't understand my grade on this assignment

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

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

Git and GitHub

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

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

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

After you run

```
git pull
```

You'll probably have to resolve a merge conflict

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

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

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

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

My command line says I cannot use a password

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

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

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

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

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

If this doesn't work, you can follow the alterntive in the porfolio readme.

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

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

My portfolio won't compile

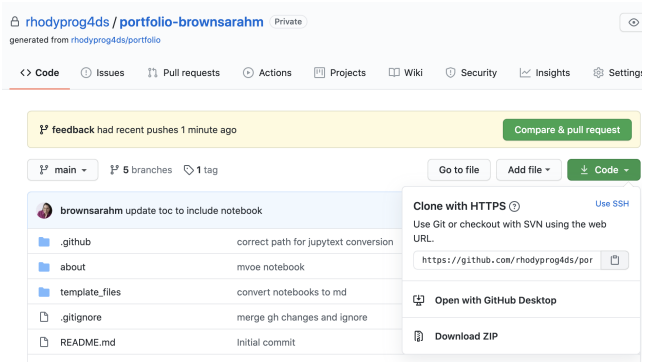
If there's an error your notebook it can't complete running. You can allow it to run if the error is on purpose by changing settings as mentioned on the formatting page.

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

That's ok. You can fix it.

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

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



Next open a terminal or GitBash and type the following.

```
git clone
```

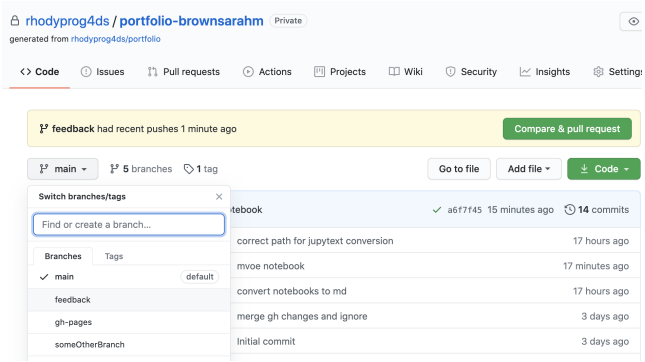
then past your url that you copied. It will look something like this, but the last part will be the current assignment repo and your username.

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

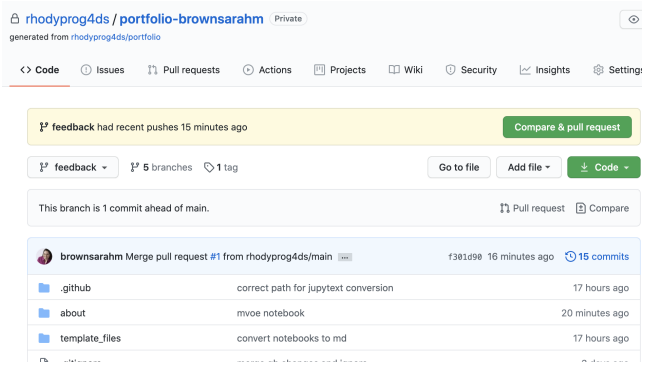
When you merged the Feedback pull request you advanced the `feedback` branch, so we need to hard reset it back to before you did any work. To do this, first check it out, by navigating into the folder for your repository (created when you cloned above) and then checking it out, and making sure it’s up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

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

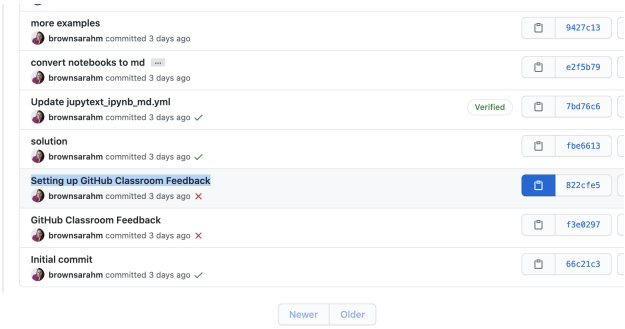
Now, you have to figure out what commit to revert to, so go back to GitHub in your browser, and swithc to the feedback branch there. Click on where it says `main` on the top right next to the branch icon and choose feedback from the list.



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



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



Now, back on your terminal, type the following

```
git reset --hard
```

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

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

If it works, your terminal will say something like

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

but the number on yours will be different.

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

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

```
git status
```

and it should show something like

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

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

Now to make GitHub match your reverted local copy.

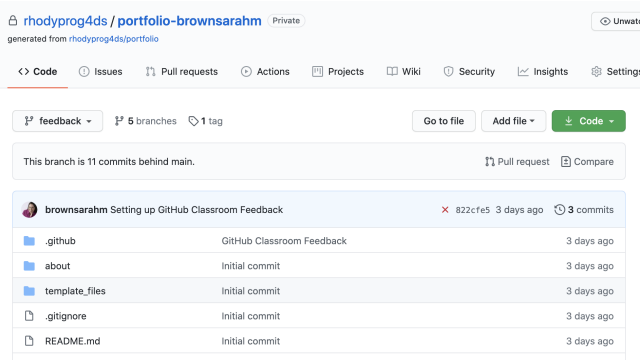
```
git push origin -f
```

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

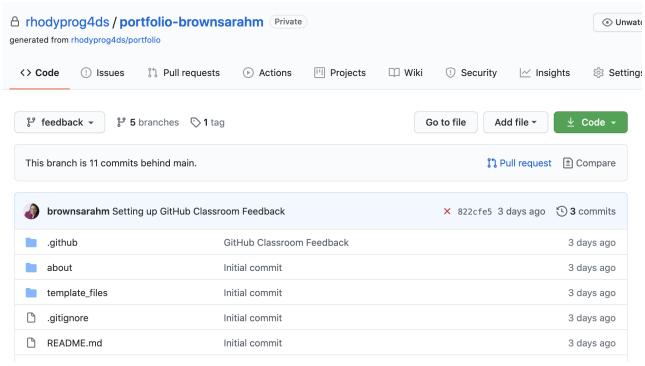
```
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
 + f301d90...822cfe5 feedback -> feedback (forced update)
```

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

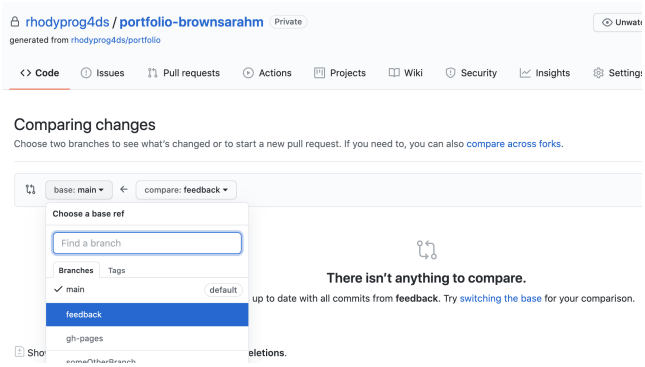
Now back on GitHub, in your browser, click on the code tab. It should look something like this now. Notice that it says, "This branch is 11 commits behind main" your number will be different but it should be 1 less than the number you had when you checked `git status`. This is because we reverted the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



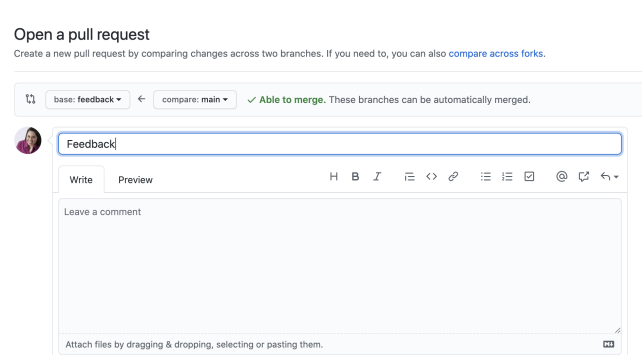
Now, you need to recreate your Pull Request, click where it says pull request.



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



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



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

Now you're done!

If you have trouble, create an issue and tag `@@rhodyprog4ds/fall22instructors` for help.

Code Errors

Key Error

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

<bound method

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

Glossary

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

aggregate

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

anonymous function

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

BeautifulSoup

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

corpus

(NLP) a set of documents for analysis

DataFrame

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

dictionary

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

document

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

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 invovlees square brackets `[]` (noun) the index of a dataframe is like a column, but it can be used to refer to the rows. It's the list of names for the rows.

interpreter

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

iterate

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

iterable

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

kernel

in the jupyter environment, the kernel is a language specific computational engine

lambda

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

PEP 8

Python Enhancement Proposal 8, the Style Guide for Python Code.

repository

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

suffix

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

Series

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

Split Apply Combine

a paradigm for splitting data into groups using a column, applying some function(aggregation, transformation, or filtration) to each piece and combinging 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 gneeral)

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
```

	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 **warning some of these require API calls and that is not recommended until at least A4**
- 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.
- 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

General Sources

These may require some more work

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

Datasets in many parts

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

Datasets with time

- Superbowl commercials

Databases

- SQLite Databases

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

General Tips and Resources

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

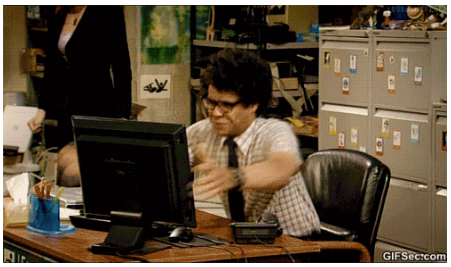
on email

- how to e-mail professors

How to Study in this class

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

Remember the goal is to avoid this:



Why this way?

Learning to program requires iterative practice. It does not require memorizing all of the specific commands, but instead learning the basic patterns. Using reference materials frequently is a built in part of programming, most languages have built in help as a part of the language for this reason. This course is designed to have you not only learn the material, but also to build skill in learning to program. Following these guidelines will help you build habits to not only be successful in this class, but also in future programming.

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

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 envrionment, for example, are *high frustration* activities, when you're learning, because you don't know what you don't know. On the other hand, diving deeper into options and more complex applications of what you have already seen in class, while challenging, is something I'm confident that you can all be successful at with minimal frustration once you've seen basic ideas in class. My goal is that you can repeat the patterns and processes we use in class outside of class to complete assignments, while acknowledging that you will definitely have to look things up and read documentation outside of class.

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

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

After class

After class, you should practice with the concepts introduced.
This means reviewing the notes: both yours from class and the annotated notes posted to the course website.
When you review the notes, you should be adding comments on tricky aspects of the code and narrative text between code blocks in markdown cells. While you review your notes and the annotated course notes, you should also read the documentation for new modules, libraries, or functions introduced that day.

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

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

Assignments

In assignments, you will be asked to practice with specific concepts at an intermediate level. Assignments will apply the concepts from class with minimal extensions. You will probably need to use help funcitons 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 problms. 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

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

Describing what you have so far

Stackoverflow is a common place for programmers to post and answer questions.
As such, they have written a good guide on creating a minimal, reproducible example.

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

Note

A fun version of this is rubber duck debugging

Understanding Errors

Error messages from the compiler are not always straight forward.

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

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

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

> **Ram Token Opportunity**
>
> If you try out friendly traceback and find it helpful, add a testimonial here. using
>
> ```
> ```{epigraph}
> ```
> ```

Terminals and Environments

Why all this work?

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

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

I think it's worth this type of pain though, because all fo the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your indepence 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.

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

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

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 portflio, 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 acount. Since your account is not the owner, they do not show on your profile.

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

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

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

Letters to Future students

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

Contributed Notes

Reviewing notes

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

Attend Office Hours

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

- David

To contribute

Via GitHub directly:

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

By Professor Sarah M Brown