About this Book

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

This class meets MWF 3-3:50pm in Chafee Social Sci Center 235.

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

> **Ram Token Opportunity**
>
> Chances to earn ram tokens are highlighted this way.

Basic Facts

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 syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly online at rhodyprog4ds.github.io/BrownFall21/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 You can view the date of changes and exactly what changes were made on the Github commit history page.

Creating an issue 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 M Brown Office hours: TBA via zoom, link on BrightSpace

Dr. Sarah M Brown is a second year 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. You can learn more about me at my website or my research on my lab site.

You can call me Professor Brown or Dr. Brown, I use she/her pronouns.

The best way to contact me is e-mail or an issue on an assignment repo. For more details, see the Communication Section

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 Prisma Chat
- Make a GitHub Account
- Install Python
- Install Git

Prisma chat

Our class link for Prisma chat is available on Brightspace. We will use this for chatting and in-class understanding checks.

On Prisma, 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 Manual

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/BrownFall21/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub Classroom

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.

> **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

Video install instructions for Anaconda:

- Windows
- Mac

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

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

Textbook

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

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 3-3:50pm in Chafee Social Sci Center 235. Every class will include participatory live coding (instructor types code while explaining, students follow along)) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

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

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

Note

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

Achievement Definitions

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

keyword	skill	Level 1	Level 2	Level 3
python	pythonic code writing	python code that mostly runs, occasional pep8 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
classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit 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	can fit linear regression models	can fit and explain regrularized or nonlinear regression
clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results
evaluate	Evaluate model performance	Explain basic performance metrics for different data science tasks	Apply basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
optimize	Optimize model parameters	Identify when model parameters need to be optimized	Manually 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

skill	Level 1	Level 2	Level 3
keyword			
unstructured	model unstructured data	Identify options for representing text data and use them once data is tranformed	Apply at least one representation to transform unstructured data for model fitting or summarizing	apply multiple representations and compare and contrast them for different end results
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred problems with a single tool pipeline	Solve semi-strucutred, completely specified problems, apply common structure to learn new features of standard tools	Scope, choose an appropriate tool pipeline and solve data science problems, 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.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
keyword														
python	1	1	0	1	1	0	0	0	0	0	0	0	0	4
process	1	1	0	0	0	0	0	0	0	1	1	0	0	4
access	0	1	1	1	0	0	0	0	0	0	0	0	0	3
construct	0	0	0	0	1	1	0	0	0	0	0	0	0	2
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
classification	0	0	0	0	0	1	1	0	0	1	0	0	0	3
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
evaluate	0	0	0	0	0	0	0	0	0	1	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
unstructured	0	0	0	0	0	0	0	0	0	0	0	1	1	2
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	4

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

Portfolios and Skills

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

	Level 3	P1	P2	P3	P4
keyword					
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	0
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	0
construct	merge data that is not automatically aligned	1	1	0	0
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	0
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters	1	1	0	0
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	0
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	0
regression	can fit and explain regrularized or nonlinear regression	0	1	1	0
clustering	apply multiple clustering techniques, and interpret results	0	1	1	0
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	0
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
unstructured	apply multiple representations and compare and contrast them for different end results	0	0	1	1
workflow	Scope, choose an appropriate tool pipeline and solve data science problems, describe strengths and weakensses of common tools	0	0	1	1

Grading

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

Principles of Grading

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

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

- Earning a C in this class means you have a general understanding of Data Science and could participate in a basic conversation about all of the topics we cover. I expect everyone to reach this level.
- Earning a B means that you could solve simple data science problems on your own and complete parts of more complex problems as instructed by, for example, a supervisor in an internship or entry level job. This is a very accessible goal, it does not require you to get anything on the first try or to explore topics on your own. I expect most students to reach this level.
- Earning an A means that you could solve moderately complex problems independently and 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. You can also earn level 1 achievements from adding annotation to a section of the class notes.

Assignments

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

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

> **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.

Portfolio Checks

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

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

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

TLDR

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

Detailed mechanics

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

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

```
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
```

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

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

```
'A'
```

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

```
'C-'
```

Or use `assert` to test it formally

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

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

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

Late work

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

> **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.

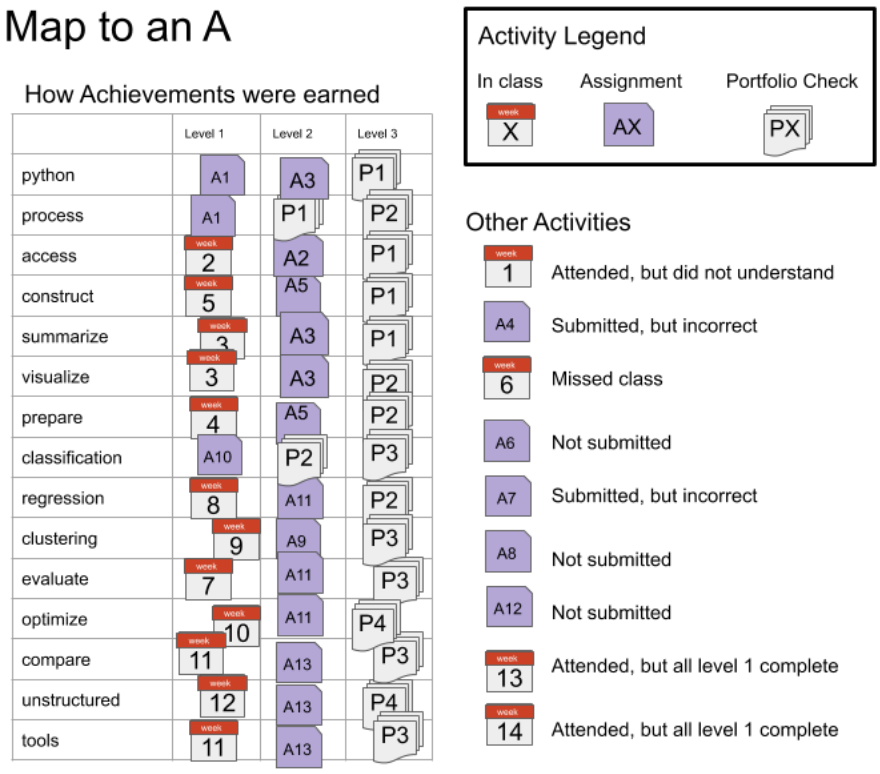
This request should be no more than 7 sentences.

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.

Grading Examples

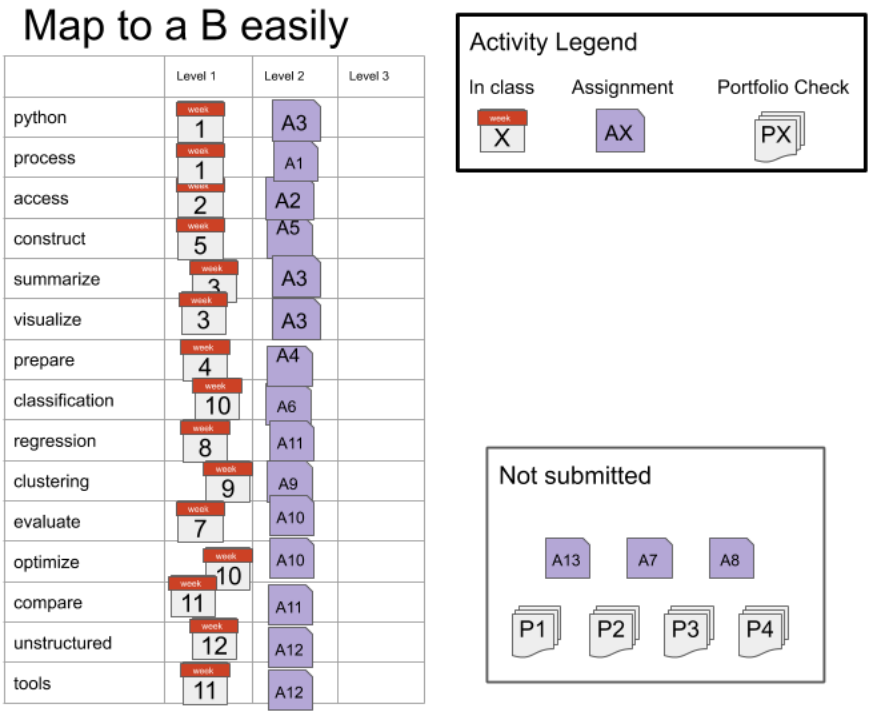
If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check or assignment 13.

Getting an A Without Perfection



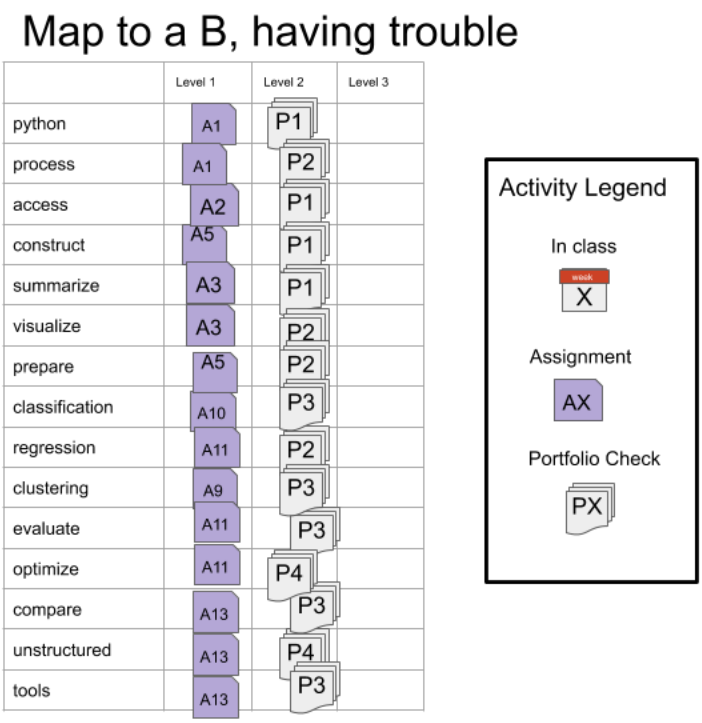
In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level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

Ram Tokens

Ram Tokens in this course will be used as a currency for extra effort. You can earn Ram Tokens by doing work that supports your learning or class activities, but do not directly demonstrate achievements. You can spend Ram Tokens to get extra grading. This will be mostly applicable to Portfolio Checks. In Checks 3 & 4, some achievements will not be eligible for grading as per the table. However, you can exchange Ram Tokens to make more achievements eligible for assessment. This system rewards you for putting in consistent effort, even if it takes you many tries to understand a concept.

To accumulate Ram Tokens, you submit a 'Deposit' to the Ram Token Bank: http://drsmb.co/ramtoken with a link to what you did to earn a token. To apply Ram tokens for extra grading, submit the same form, with a link to the assignment and add the Ramtoken label to the Feedback PR.

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.

Policies

Anti-Bias Statement:

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

Disability Services for Students Statement:

Your access in this course is important. Please send me your Disability Services for Students (DSS) accommodation letter early in the semester so that we have adequate time to discuss and arrange your approved academic accommodations. If you have not yet established services through DSS, please contact them to engage in a confidential

conversation about the process for requesting reasonable accommodations in the classroom. DSS can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dss@etal.uri.edu. We are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

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

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

URI COVID-19 Statement

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

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

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

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

Course Communications

Help Hours

Day	Time	Location	Host
Monday	9:30:00 AM-10:30 AM	inperson Tyler Hall 140	Chamudi
Monday	12:30:00 PM-2:00 PM	inperson Tyler Hall 139	Chamudi
Wednesday	4:00:00 PM-5.00 PM	inperson Tyler Hall 139	Chamudi
Wedneday	1:30:00 PM-3:00 PM	inperson Tyler Hall 140	Chamudi
Wednesday	7:00:00 PM-8:30	gather.town	Sarah
By appointment	scheduling link on Brightspace	in person Tyler 134	Sarah

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

To reach out, By usage

usage	platform	area	note
in class	prismia	chat	outside of class time this is not monitored closely
any time	prismia	message board	for discussion with peers
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	github	pull request on website	remember to request ram tokens if applicable
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	pull request on website	remember to request ram tokens if applicable

Use prismia for

usage	area	note
in class	chat	outside of class time this is not monitored closely
any time	message board	for discussion with peers
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

Today's goals:

1. Operate tools for in-class participation
2. Understand what Data Science is, in broad terms
3. Understand the syllabus (grading, topics covered, schedule, etc)
4. Understand how to learn in this course

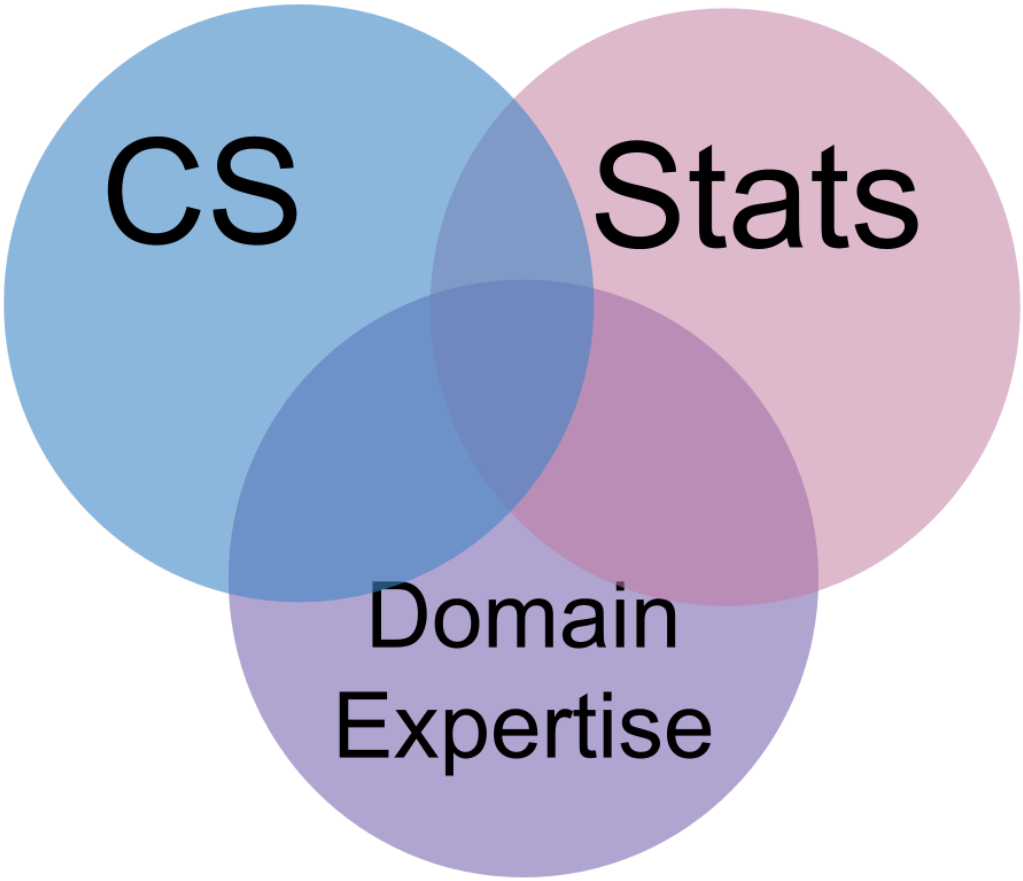
1.1. Prismia Chat

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

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

1.2. What is Data Science

In general:

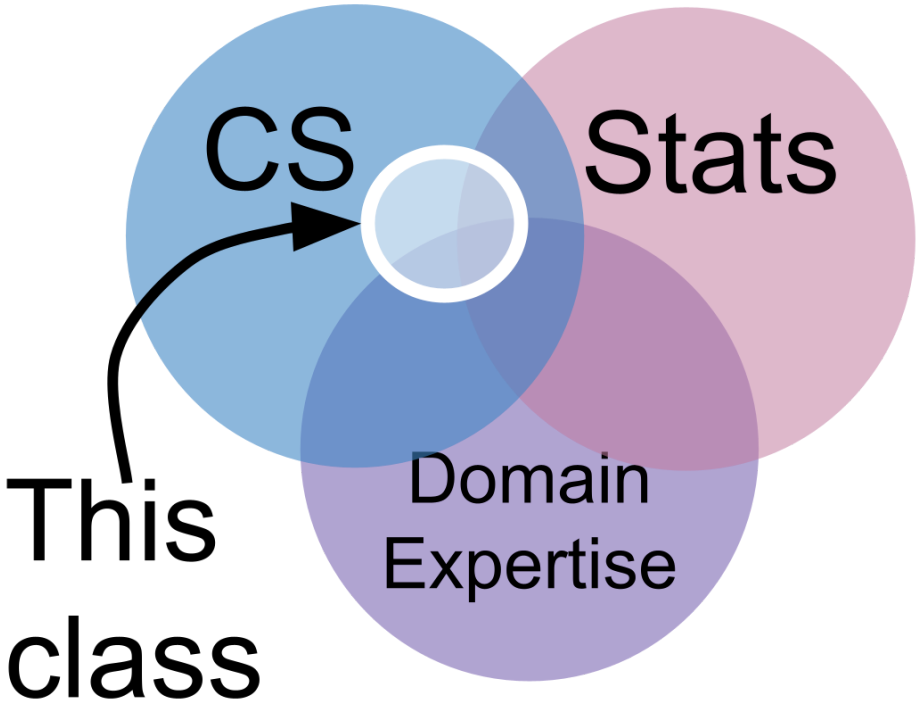


statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

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.

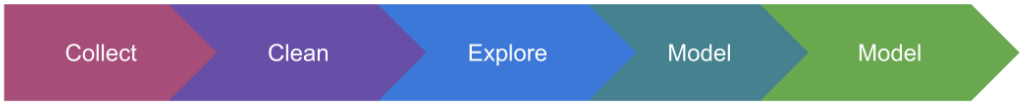
For this class



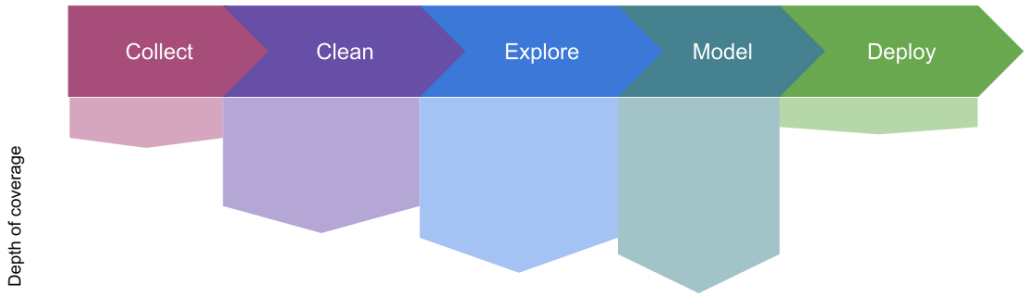
We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

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

1.2.1. How does data science happen?

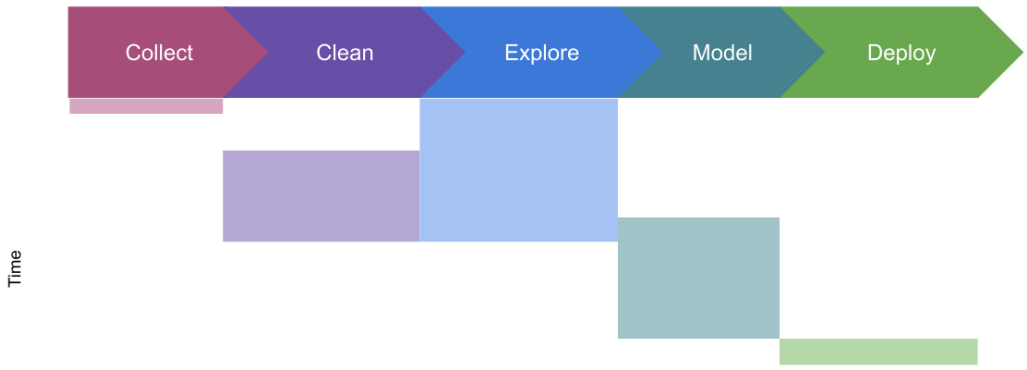


1.2.2. how we'll cover it, in depth



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

1.2.2.1. 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 cleaneed the data.

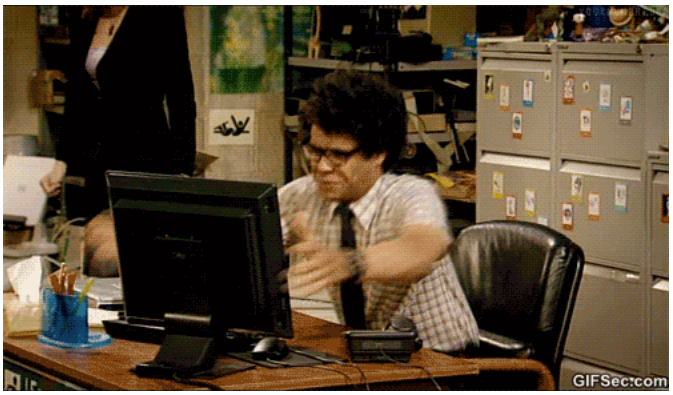
1.3. How this class will work

- today is an exception
- in general we'll be live coding

Let's look at the syllabus

Read carefully to make sure you understand the grading; it's not typical points and an average.

Class is designed to avoid this:

1.4. 

1.5. Learning Cycle

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

1.6. 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. Jupyter Notebook Tour & Python Review

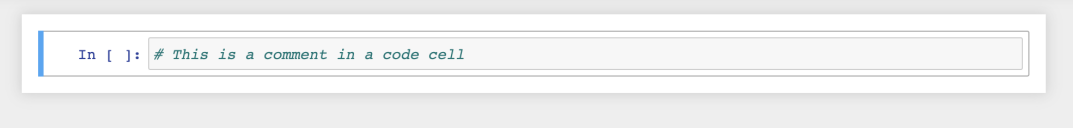
2.1. A jupyter notebook tour

Launch a `jupyter notebook`:

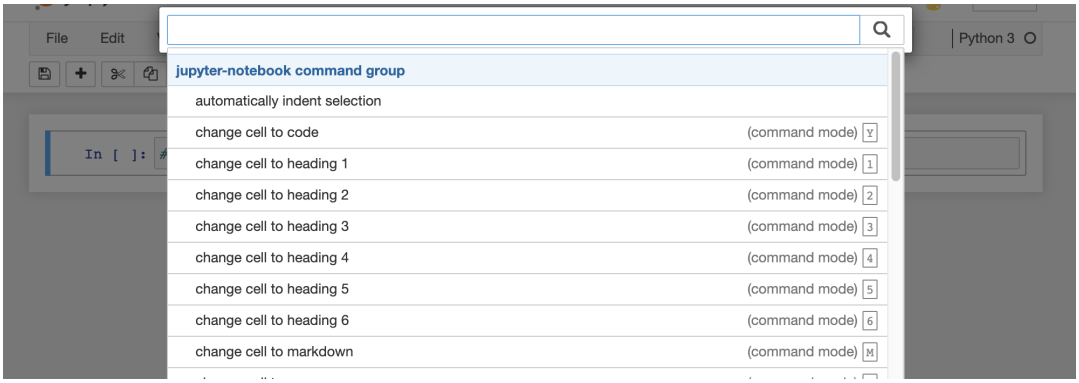
- on Windows, use anaconda terminal
- on Mac/Linux, use terminal

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

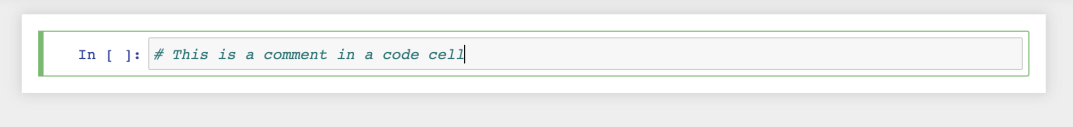
A Jupyter notebook has two modes. When you first open, it is in command mode. The border is blue in command mode.



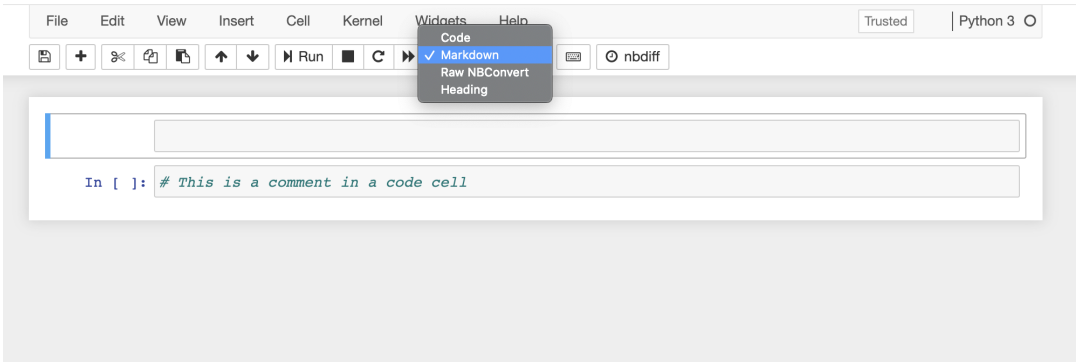
When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.



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



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



++

This is a markdown cell

- we can make
- itemized lists of
- bullet points

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

2.1.1. 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'
```

Note

For a little more python review, see my 2020 CSC310 notes this is just enough for this assignment.

2.2. Just enough Git for Assignment 1

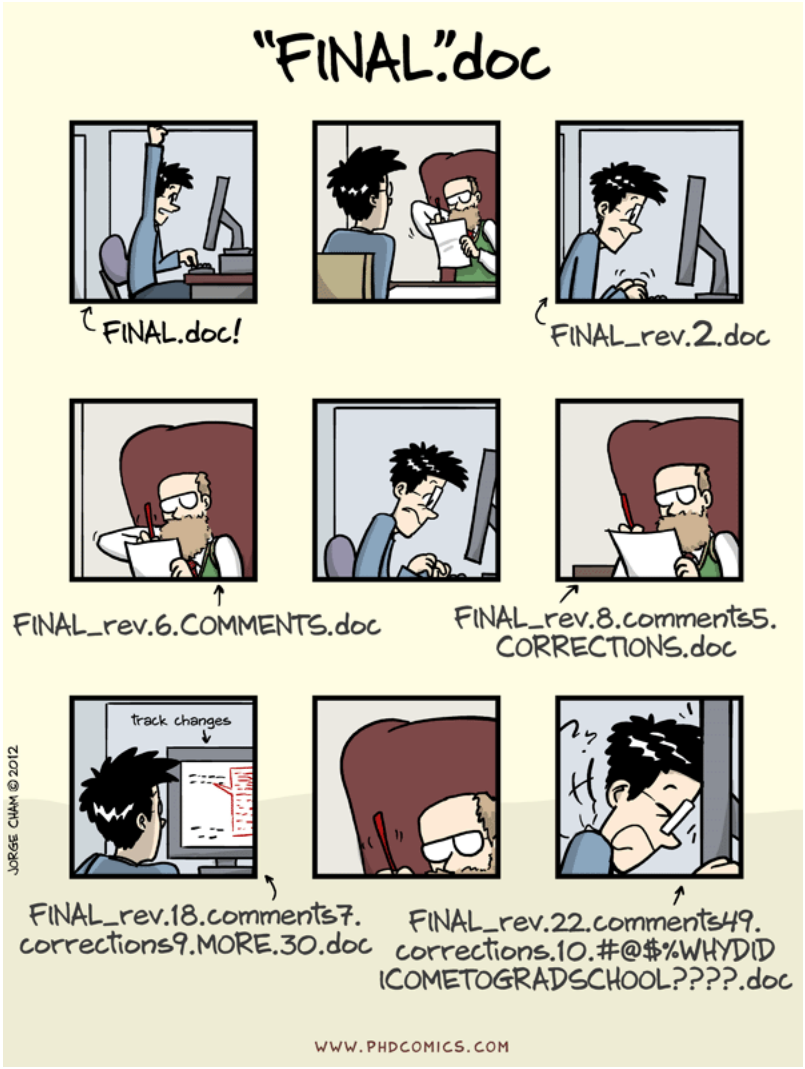
2.2.1. Assignment 1:

Goals for this assignment

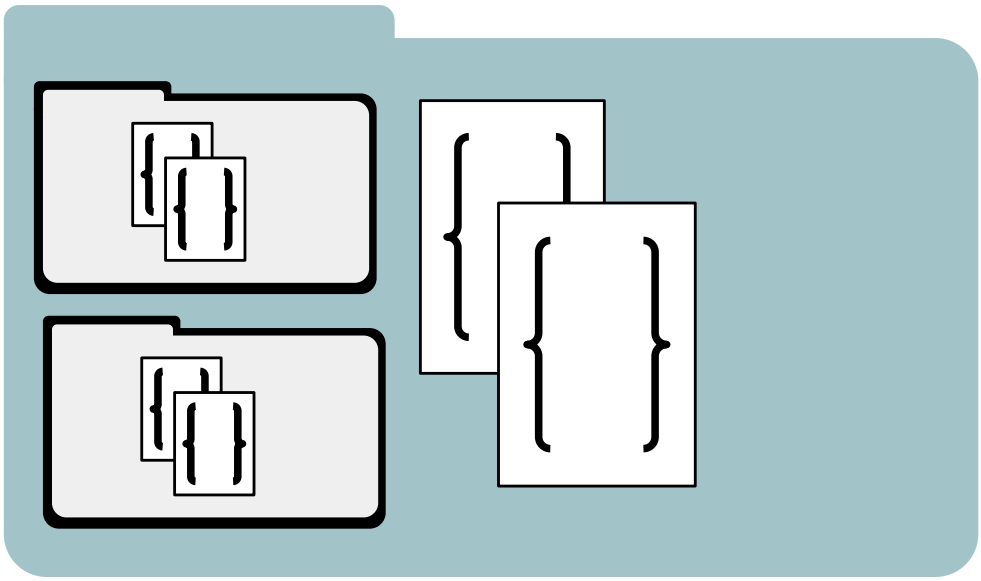
- setup your portfolio
- check that you understand the grading
- review Python basics
- practice with git and GitHub

2.2.2. Why Version control

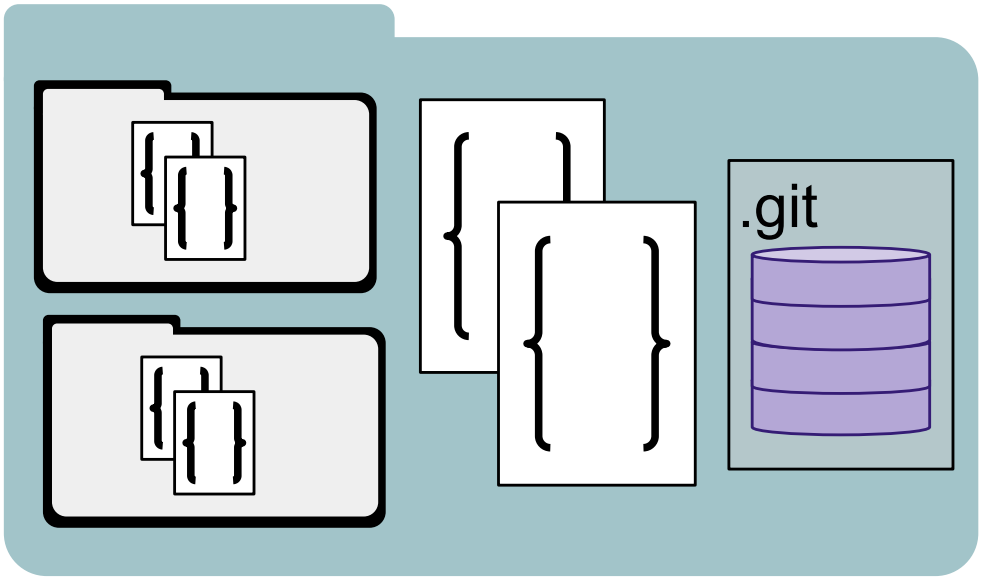
We often want to keep track of the different versions in case we want to go back, but this can be painful:



We typically organize projects in folder

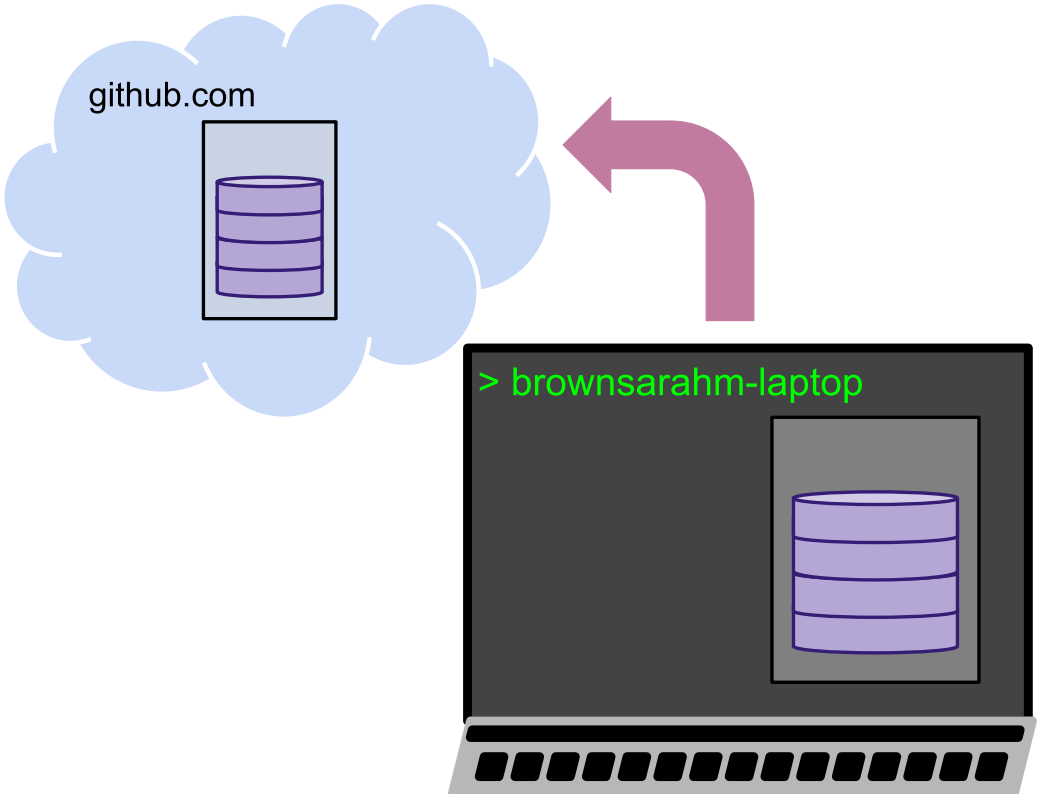


A repository is a folder with a hidden directory named `.git`

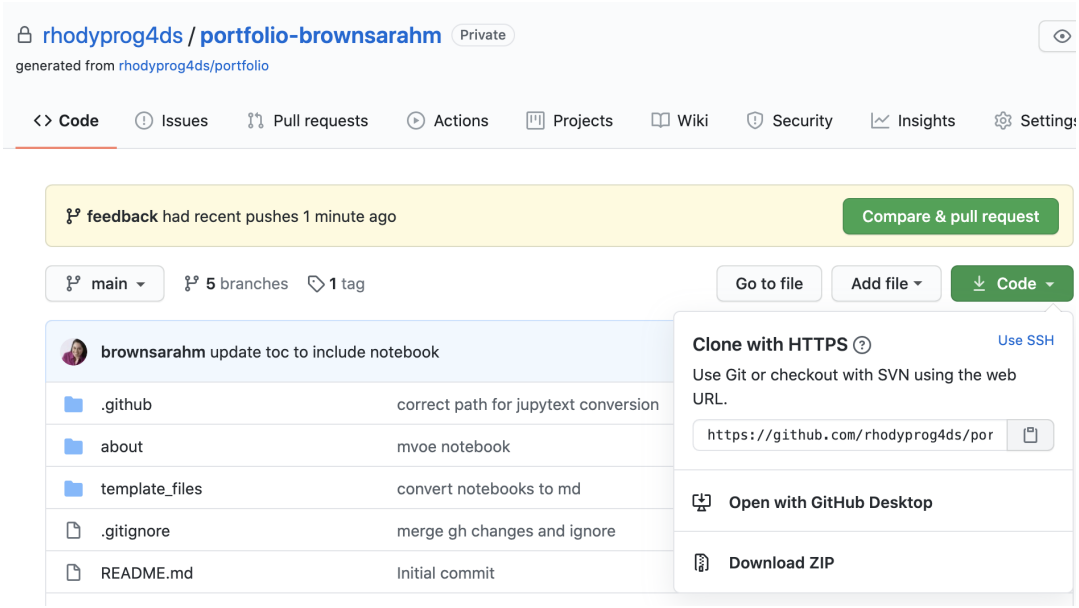


The `git` applicaiton manages that hidden directory, we don't write to it directly, which is why we keep it hidden.

Git is a distributed system, you have a local version and a remote version.



Once a repository exists on GitHub, we get a local copy by cloning it after we get its address from the GitHub interface, by clicking on the green code butto that is below the menu area to the right. It's at the top right corner of the list of files in the repository.



For this part, use GitBash on windows or terminal otherwise: If you set up a Personal Access Token you can use the https version

After `cd/to/where/you/want/your/repo/locally`:

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

If you set up ssh keys you use that instead

```
git clone git@github.com:rhodyprog4ds/portfolio-example.git
```

Once it's cloned, then you can navigate into the new folder:

```
cd portfolio-example
```

Then you can change files, for example adding to the intro.

Some common actions in Git, you'll want.

Check on the status of your repository:

```
git status
```

Add files to the staging area:

```
git add filename
```

Add all changes to the staging area:

```
git add .
```

Commit your changes to the repository:

```
git commit -m 'a message that will help your future self know what this part is'
```

Push your changes to GitHub

```
git push
```

Pull changes from GitHub

Note

These notes can be downloaded as an actual notebook, click the GitHub logo at the top of the page and choose .ipynb. The following is not runnabel in the notebook as is.

```
git pull
```

You can also go through these same basic steps: add, commit, push

2.3. More on git

- GitHub Hello World
- Software Carpentry Git Novice Lesson

Also, in Spring 2022, I'm teaching a section of CSC392: Topics in Computing, Introduction to Computer Systems, that will cover tools of the trade (git, bash, etc) and how they all work in great detail.

2.4. More on Python

Read Pep 8 to see what good style in Python is.

3. Getting help, object inspection, loading data

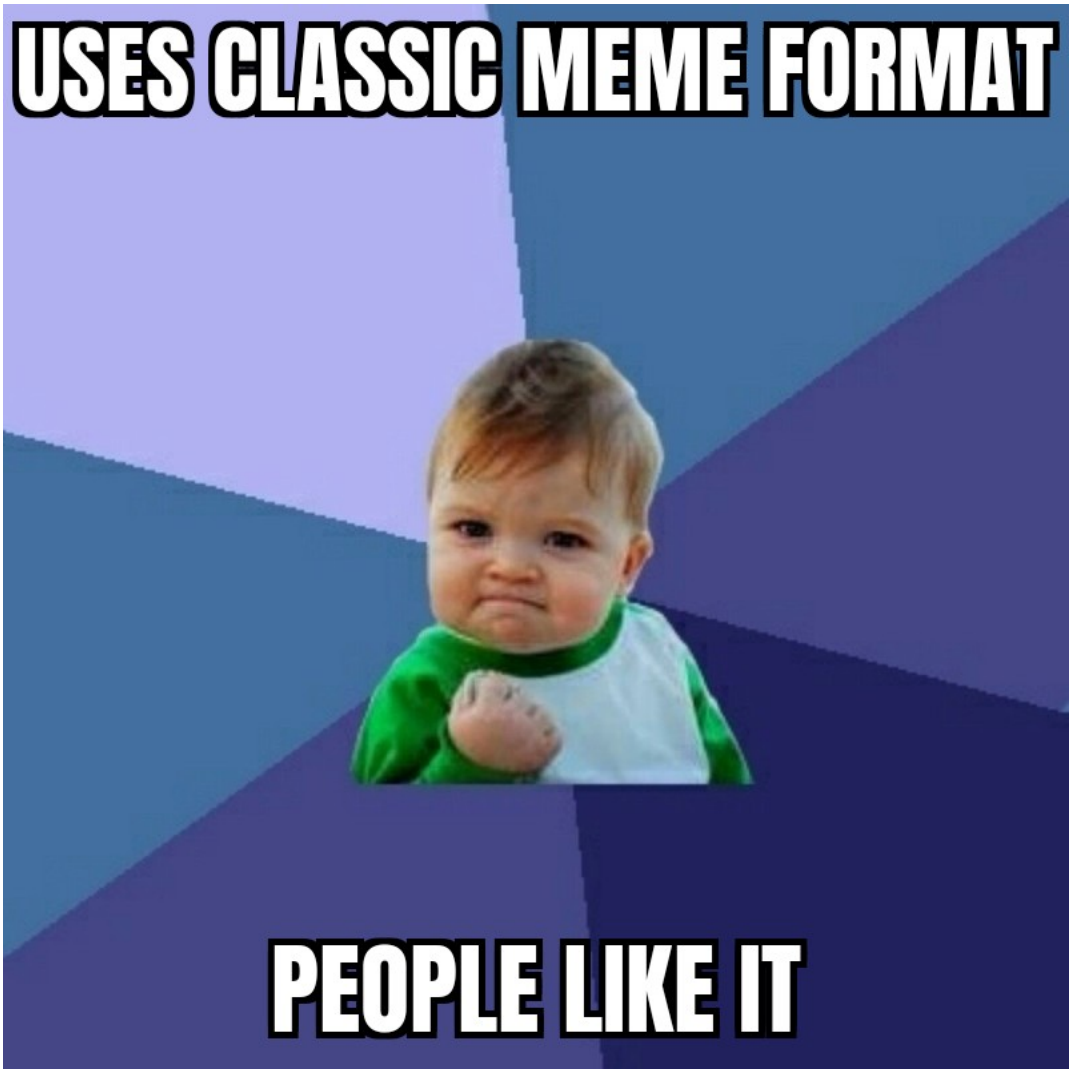
3.1. First, Don't Worry members

Class Response Summary:

funny CS memes


Hey, Don't even worry about it.


David Steele
@David_C_Steele
"We just left a rest stop! Why didn't you go then?"
SMITH



3.2. Getting Help in Jupyter

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 use jupyter popup help wiht shift +tab or `?`

```
print?
```

Or the base python `help` function

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

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

Notice that function can take multiple arguments and has a keyword argument (must be used like `argument=value`) described as `sep=' '`. This means that by default it adds a space

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

```
Sarah Brown
```

But we can change the separator.

```
Sarah_Brown
```

Note that it also defaults to end to use `\n`

```
print(first,last)
print('hello')
```

```
Sarah Brown
hello
```

Where does this help information come from?

Note

You can copy code from the notes, try hovering over this

```
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
```

We can apply `help` on the function we wrote

```
help(compute_grade)
```

```
Help on function compute_grade in module __main__:

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 (+/-)
```

It gets the docstring

3.3. Everything is an Object in Python

we can use the builtin function `type` to inspect them, and get attributes with `.`

```
type(compute_grade)
```

```
function
```

```
compute_grade.__name__
```

```
'compute_grade'
```

```
c = 4.5
```

```
type(c)
```

```
float
```

```
c= 'hello'
```

```
type(c)
```

```
str
```

When do we use single vs double quotes?

- You can use either, unless you need to put one inside the string then use the other.

```
my_sentence = "The professor's name is Dr. Brown"
```

```
my_sentence = 'The professor's name is Dr. Brown'
```

```
  File "/tmp/ipykernel_1653/607286316.py", line 1
    my_sentence = 'The professor's name is Dr. Brown'
                                 ^
SyntaxError: invalid syntax
```

Yes we can escape special characters:

```
my_sentence = 'The professor\'s name is Dr. Brown'
```

but, it's less readable and not recommended.

3.4. Good Code is always relative

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. Coffee Data

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

How was this dataset collected?

- reviewrs added to DB
- then scraped

Where did it come from?

- offee Quality Institute's trained reviewers.

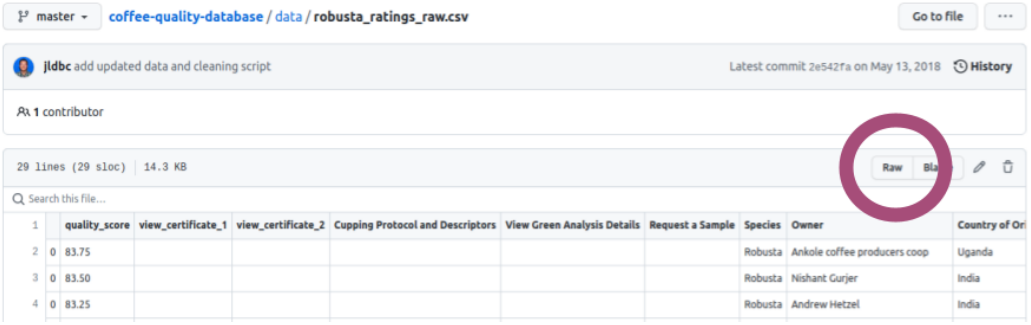
what format is it provided in?

- csv (Comma Separated Values)

what other information is in this repository?

- the code to scrape

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



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

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

We will use a library called Pandas

```
import pandas as pd
# import library and give it an alias (nickname) pd
```

```
pd.read_csv(data_url)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ank cof produc
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuram est
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	N
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	uga
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katu developm tr
5	6	Robusta	andrew hetzel	India	NaN	NaN	(s
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuram est
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuram est
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubu un
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	uga
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuram est
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	N
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ank cof produc coop uni
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	N
16	17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuram esta
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawac
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaa
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	manr cof proj
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuram esta
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuram esta

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	o laborat
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	ca cazen
26	27	Robusta	cafe politico	India	NaN	NaN	N
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	N

28 rows × 44 columns

Try it yourself

Read the data in again, but with the index correct and save it to a variable.

Once we read it in, we van view the first 5 rows with the `head` method.

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.N
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	

5 rows × 43 columns

Important

Remember to comment & annotate your code

3.6. Follow Up questions

3.6.1. General Questions

How do you create code to scrape data from a website and compile it into a csv file?

Will we be using pandas a lot during the semester?

3.6.2. Clarifying

How do you auto finish your directories

How do you properly shut down Jupyter Notebook

Is pd some sort of variable we set or was it built in?

How should I be organized for this class? Keep it all in a single folder? Keep it on GitHub?

I'm still not sure how to keep everything together in a portfolio for the semester?

I am still wondering if I am using anaconda or just normal terminal

Can I push this code into my portfolio using the anaconda terminal

3.6.3. Grading Questions

How do we keep track of which achievements we've earned?

I don't really have many questions from today, but I was wondering if office hours were posted.

Will we always submit homework through the portfolio folder in github?

I'm just confused as how to view my feedback from the assignment

3.6.4. Questions we'll answer later this week

- does each column have a number assigned to it in data frames?
- Can other data types be imported into a notebook and edited the same way as .csv files?

3.7. More Practice

- How could you check if `pd` is built in or if we defined it?
- If we wanted to see more than 5 rows when printing the head of the dataset how would we do so?

> **Ram Token Opportunity**
>
> Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

4. Pandas DataFrames

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

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

4.1. More about loading libraries

We can import pandas without the alias `pd` if we want, but then we have to use the full name everywhere

```
import pandas
```

```
pandas.read_csv(coffee_data_url)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ank cof produc
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuram est
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	N
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	uga
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katu developm tr
5	6	Robusta	andrew hetzel	India	NaN	NaN	(s
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuram est
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuram est
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubu un
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	uga
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuram est
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	N
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ank cof produc coop uni
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	N
16	17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuram esta
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawac
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaa
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	manr cof proj
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	N
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuram esta
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuram esta

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	o laborat
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	ca cazen
26	27	Robusta	cafe politico	India	NaN	NaN	N
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	N

28 rows × 44 columns

We'll use `pd` because that's the more common convention and so that we can type fewer characters throughout our code

```
import pandas as pd
```

4.2. Examining DataFrames

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

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

```
df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.N
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	

5 rows × 43 columns

Using help, we can see that that head takes one parameter and has a default value of 5, which is why we got 5 rows, but we can get 2 instead

```
df.head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Num
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017

2 rows × 43 columns

We can look at the last rows with `tail`

```
df.tail(3)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	C
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	opp
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	

3 rows × 43 columns

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

```
type(df)
```

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

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

```
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')
```

These are a special type called Index

```
type(df.columns)
```

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

It also has an index

```
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')
```

and values

```
df.values
```

```
array([['Robusta', 'ankole coffee producers coop', 'Uganda', ..., 1488.0,
        1488.0, 1488.0],
       ['Robusta', 'nishant gurjer', 'India', ..., 3170.0, 3170.0,
        3170.0],
       ['Robusta', 'andrew hetzel', 'India', ..., 1000.0, 1000.0, 1000.0],
       ...,
       ['Robusta', 'james moore', 'United States', ..., 795.0, 795.0,
        795.0],
       ['Robusta', 'cafe politico', 'India', ..., nan, nan, nan],
       ['Robusta', 'cafe politico', 'Vietnam', ..., nan, nan, nan]],
      dtype=object)
```

it also knows its own shape

```
df.shape
```

```
(28, 43)
```

we can use builtin fucntions on our DataFrame too not just its own methods and attributes.

```
len(df)
```

```
28
```

Why does `len` turn green? it's a python reserve word

4.3. Building a Data Frame programmatically

One way to build a data frame is from a dictionary:

```
people = {'names':['Sarah','Connor','Kenza'],
          'username':['brownsarahm','sudoPsych','kbdlh']}
```

```
people
```

```
{'names': ['Sarah', 'Connor', 'Kenza'],
 'username': ['brownsarahm', 'sudoPsych', 'kbdlh']}
```

```
type(people)
```

```
dict
```

```
people_df = pd.DataFrame(people)
people_df
```

	names	username
0	Sarah	brownsarahm
1	Connor	sudoPsych
2	Kenza	kbdlh

```
type(people['names'])
```

```
list
```

```
type(people)
```

```
dict
```

```
type({4,5,5})
```

```
set
```

```
{4,5,5}
```

```
{4, 5}
```

```
people['names']
```

```
['Sarah', 'Connor', 'Kenza']
```

```
type(set(people['names']))
```

```
set
```

```
unique_people = set(people['names'])
type(unique_people)
```

```
set
```

```
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')
```

```
for col in df.columns:
    print(col.split('.'))
```

```
['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']
```

```
for key,value in people.items():
    print(key,':',value)
```

```
names : ['Sarah', 'Connor', 'Kenza']
username : ['brownsarahm', 'sudoPsych', 'kbdlh']
```

```
df['Owner']
```

```
1          ankole coffee producers coop
2                       nishant gurjer
3                        andrew hetzel
4                               ugacof
5          katuka development trust ltd
6                        andrew hetzel
7                        andrew hetzel
8                       nishant gurjer
9                       nishant gurjer
10                              ugacof
11                              ugacof
12                      nishant gurjer
13                       andrew hetzel
14    kasozi coffee farmers association
15         ankole coffee producers coop
16                       andrew hetzel
17                       andrew hetzel
18                  kawacom uganda ltd
19                       nitubaasa ltd
20                mannya coffee project
21                       andrew hetzel
22                       andrew hetzel
23                       andrew hetzel
24                         luis robles
25                         luis robles
26                         james moore
27                       cafe politico
28                       cafe politico
Name: Owner, dtype: object
```

```
df.Owner
```

```
1          ankole coffee producers coop
2                       nishant gurjer
3                        andrew hetzel
4                               ugacof
5          katuka development trust ltd
6                        andrew hetzel
7                        andrew hetzel
8                       nishant gurjer
9                       nishant gurjer
10                              ugacof
11                              ugacof
12                      nishant gurjer
13                       andrew hetzel
14    kasozi coffee farmers association
15         ankole coffee producers coop
16                       andrew hetzel
17                       andrew hetzel
18                  kawacom uganda ltd
19                       nitubaasa ltd
20                mannya coffee project
21                       andrew hetzel
22                       andrew hetzel
23                       andrew hetzel
24                         luis robles
25                         luis robles
26                         james moore
27                       cafe politico
28                       cafe politico
Name: Owner, dtype: object
```

```
df
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	
6	Robusta	andrew hetzel	India	NaN	NaN	(self)	
7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	
8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148
9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148
10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	
11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148
13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	
14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	
15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	
16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	
17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	
18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	
19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	
20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	
21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	
22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	
23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-11
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	

28 rows × 43 columns

Key points:

write three things to remember from today's class

4.4. Questions After Classroom

many overlapping questions today

4.5. General

How to know which function to use in certain problems or situations

4.6. Clarifying

Is there a way to have a set show the duplicates that get discarded?

being able to access the code somewhere without asking to scroll would be nice

4.7. Course Admin

When will homeworks be posted/due typically?

4.8. Questions we'll answer later

can you use cast a pandas dataframe into a set?

4.9. Try it yourself

- Create variables of three different types with facts about yourself. Use descriptive variable names relative to the contents, not their types.

- Create a list, again with a descriptive name, and print out the types

```
<class 'str'>
<class 'int'>
<class 'list'>
```

- Write a function, `type_extractor` that takes a list and a type and returns the item of that type from the list
- Test your function on all three items from your dictionary.
- Use one type of jupyter help on your function, what does it display? If it doesn't display anything modify your function so that help will work.
- Make yourself notes in the most memorable way for you about what a DataFrame is.

Ram Token Opportunity

Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

5. More Loading Data, Indexing, and Iterables

As always, we'll start with loading pandas.

```
import pandas as pd
```

5.1. Checking in on help hours

if you missed class, check over the office hours schedule and e-mail if you can or cannot attend at least one time

5.2. Portfolio Preparation and Maintainance

We'll spend a little time today getting your portfolio ready for the first check.

5.2.1. Access your portfolio

Go to your portflio

- from the course organization
- from the list of your recent repositories on the left hand side of the GitHub home page

optionally, open it locally as well (we're going to update content and)

5.2.2. Start your Know, Want to Know, Learned Table

In each portfolio submission introduction, you'll reflect on what you've learned. To get ready for that, we'll first make note of what you already know and what you want to know.

1. edit `submission_1_intro` in your portfolio locally or on GitHub:
2. In the KWL section in the first two bullets after each skill with what you know and want to know. You can edit these in more detail later.

This will render as a table in your built portflio, for reference on the syntax, refer to the Tables section of the jupyterbook Myst Markdown cheatsheet.

Warning

If you work on this in the GitHub website, be sure to pull these chances locally before you start working offline next

5.2.3. Merge the setup work

Once you're done, Go to your pull request tab, and select the feedback Pull Reques. Commit any suggestions if you'd like and then merge the PR.

Warning

only do this after grading

Note

To view the feedback, after merging the PR, remove `is:open` from the search bar on the PR page

5.3. Indexing

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

What will topics[-1] return?

```
topics[-1]
```

```
'pandas'
```

Using negative indices starts from the right. The last element is -1. The first is 0.

5.4. Reading DataFrames from Websites

We'll first read from the course website.

```
course_comms_url =
'https://rhodyprog4ds.github.io/BrownFall21/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 multple tables on the page, so lets see what it t does:

```
pd.read_html(course_comms_url)
```

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.

```
[             Day                            Time                 Location  \
 0          Monday             9:30:00 AM-10:30 AM  inperson Tyler Hall 140
 1          Monday             12:30:00 PM-2:00 PM  inperson Tyler Hall 139
 2       Wednesday              4:00:00 PM-5.00 PM  inperson Tyler Hall 139
 3        Wedneday              1:30:00 PM-3:00 PM  inperson Tyler Hall 140
 4       Wednesday                 7:00:00 PM-8:30              gather.town
 5  By appointment  scheduling link on Brightspace      in person Tyler 134

       Host
 0  Chamudi
 1  Chamudi
 2  Chamudi
 3  Chamudi
 4    Sarah
 5    Sarah  ,
                                      usage platform  \
 0                                  in class  prismia
 1                                  any time  prismia
 2                                  any time  prismia
 3         private questions to your assignment   github
 4   for general questions that can help others   github
 5                           to share resources   github
 6  matters that don't fit into another category   e-mail

                      area                                              note
 0                    chat  outside of class time this is not monitored cl...
 1            message board                          for discussion with peers
 2       download transcript  use after class to get preliminary notes eg if...
 3  issue on assignment repo                             eg bugs in your code"
 4   issue on course website  eg what the instructions of an assignment mean...
 5   pull request on website       remember to request ram tokens if applicable
 6    to brownsarahm@uri.edu  remember to include `[CSC310]` or `[DSP310]` (...  ,
                                      usage                      area  \
 0  matters that don't fit into another category  to brownsarahm@uri.edu

                                          note
 0  remember to include `[CSC310]` or `[DSP310]` (...  ,
                                    usage                      area  \
 0       private questions to your assignment  issue on assignment repo
 1  for general questions that can help others   issue on course website
 2                         to share resources   pull request on website

                                          note
 0                            eg bugs in your code"
 1  eg what the instructions of an assignment mean...
 2       remember to request ram tokens if applicable  ,
      usage                area  \
 0  in class                chat
 1  any time        message board
 2  any time  download transcript

                                          note
 0  outside of class time this is not monitored cl...
 1                         for discussion with peers
 2  use after class to get preliminary notes eg if...  ]
```

It appears to have read all of them, lets check the type:

```
type(pd.read_html(course_comms_url))
```

```
list
```

Since we know it's a list, we'll save it to a variable that indicates that.

```
comms_list = pd.read_html(course_comms_url)
```

If we get just the first element,

```
type(comms_list[0])
```

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

it's a DataFrame and prints accordingly.

```
comms_list[0]
```

	Day	Time	Location	Host
0	Monday	9:30:00 AM-10:30 AM	inperson Tyler Hall 140	Chamudi
1	Monday	12:30:00 PM-2:00 PM	inperson Tyler Hall 139	Chamudi
2	Wednesday	4:00:00 PM-5.00 PM	inperson Tyler Hall 139	Chamudi
3	Wedneday	1:30:00 PM-3:00 PM	inperson Tyler Hall 140	Chamudi
4	Wednesday	7:00:00 PM-8:30	gather.town	Sarah
5	By appointment	scheduling link on Brightspace	in person Tyler 134	Sarah

Since it's a list, we can use base python's `len` function to check how many tables there are

```
len(comms_list)
```

```
5
```

We've seen the first table and know it's the help hours, so we can save that to a separate variable and use it

```
help_df = comms_list[0]
```

We've inspected the dataframe some before, but we can also check the type of each column.

```
help_df.dtypes
```

```
Day         object
Time        object
Location    object
Host        object
dtype: object
```

Further Reading

You can read more about the details of data types in Pandas in the documentation

5.5. How are objects printed in jupyter?

Question from class

Q: Why does it have `dtype:object` after the type for each row? A: the last line is information about the object that is being printed out.

To understand this, let's save the thing we're curious to a variable so we can examine it multiple ways more easily.

Note

this section is added, it didn't happen this way in class. This section, describes **how** I figured out the answer to the question about why that extra line is diplayed.

```
help_df_types = help_df.dtypes
```

Next we'll check the type of this object and its shape

```
type(help_df_types)
```

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

a Series is like a DataFrame, but just one row with headings, and then rotated.

```
help_df_types.shape
```

```
(4,)
```

This means that it's length is 4 and it's a 1 dimensional object; the column headers have converted to an index and are treated as metadata, but not a part of the actual data.

So, the line we're interested in is not a part of the object, because it's length 4 and the thing we're curious about is the fifth line.

We'll pick one variable from the DataFrame and check its type

```
type(help_df['Day'])
```

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

This is also a Series, so lets check its output

```
help_df['Day']
```

```
0            Monday
1            Monday
2         Wednesday
3          Wedneday
4         Wednesday
5    By appointment
Name: Day, dtype: object
```

THe last line of this one is information about the Series, its name, and its dtype.

Let's make another series, and see how it prints

```
pd.Series([5,4,5])
```

```
0    5
1    4
2    5
dtype: int64
```

The last line is the dtype of the Series; so in our original object, that last line is because the list of dtypes is the of type object.

```
help_df_types
```

```
Day         object
Time        object
Location    object
Host        object
dtype: object
```

5.6. How do we know what to check?

we examined the DataFrame so far by (me) knowing what to look for.

In python objects you can progrmamatically find what to look for with the `__dict__` attribute or we can rely on the online documentation or use it via help.

In ipython (what we use in jupyter, by default) we can use the `?` for help

```
pd.DataFrame?
```

```
help(pd.DataFrame)
```

Everything is Data

writing good documentation lets people who use your code get help for free Not only do help tools use the docs, but that website is generated programmatically using a tool called Sphinx from the documentation inside the code. You can access the docstring of a python using the `.__doc__` attribute

Caution

ipython help read about how it works, if it doesn't work for you to try to figure out why

```
Help on class DataFrame in module pandas.core.frame:

class DataFrame(pandas.core.generic.NDFrame, pandas.core.arraylike.OpsMixin)
 |  DataFrame(data=None, index: 'Axes | None' = None, columns: 'Axes | None' = None,
dtype: 'Dtype | None' = None, copy: 'bool | None' = None)
 |
 |  Two-dimensional, size-mutable, potentially heterogeneous tabular data.
 |
 |  Data structure also contains labeled axes (rows and columns).
 |  Arithmetic operations align on both row and column labels. Can be
 |  thought of as a dict-like container for Series objects. The primary
 |  pandas data structure.
 |
 |  Parameters
 |  ----------
 |  data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
 |      Dict can contain Series, arrays, constants, dataclass or list-like objects. If
```

```
|      data is a dict, column order follows insertion-order.
|
```

Try it yourself!

Make a list of the shape of all of the tables on the syllabus Achievements page.

```
achievements_url =
'https://rhodyprog4ds.github.io/BrownFall21/syllabus/achievements.html'
```

```
[(14, 3), (15, 5), (15, 15), (15, 6)]
```

This solution uses a list comprehension which allows us to compress a loop. It's equivalent to the following with a for loop

```
[(14, 3), (15, 5), (15, 15), (15, 6)]
```

5.7. Lambdas and Dictionaries for switching

What if we want to print out the first column for the dataFrame if it has more than 3 columns and the whole thing if it has 3 or less columns?

Two ways of writing a function

```
# with the def key
def first_col_f(d):
    return d[d.columns[0]]

# lambda (anonymous function)
first_col_l = lambda d: d[d.columns[0]]

first_col_f(help_df) == first_col_l(help_df)
```

```
0    True
1    True
2    True
3    True
4    True
5    True
Name: Day, dtype: bool
```

Try it yourself

read the code excerpt above carefully and try to match up the parts of a function: its name, the parameter list, and the body. Try writing your own lambda function.

Lambdas are an example of an anonymous function

We can put functions in dictionaries, or even define a lambda right in the dictionary.

```
df_display = {True: lambda d: d[d.columns[0]],
       False: lambda d: d}

for df in pd.read_html(achievements_url):
    _, n_cols = df.shape
    print(df_display[n_cols>3](df))
```

Question from class

Python does have ternary operators but the dictionary is a more common way to achieve this and goal and more common patterns are better for readability

Further Reading

You can refer to the official documentation on lambda functions for a brief syntax description or this tutorial on Real Python for more context, history, and examples.

Important

We'll see lambdas again and again. Starting with summarizing data and again when cleaning. Understanding how to use these will help.

```
   Unnamed: 0_level_0                                              topics  \
                 week                                 Unnamed: 1_level_1
0                   1                              [admin, python review]
1                   2                         Loading data, Python review
2                   3                           Exploratory Data Analysis
3                   4                                       Data Cleaning
4                   5                       Databases, Merging DataFrames
5                   6  Modeling, Naive Bayes, classification performa...
6                   7                   decision trees, cross validation
7                   8                                          Regression
8                   9                                          Clustering
9                  10                              SVM, parameter tuning
10                 11                              KNN, Model comparison
11                 12                                       Text Analysis
12                 13                                     Images Analysis
13                 14                                       Deep Learning

                               skills
                   Unnamed: 2_level_1
0                             process
1      [access, prepare, summarize]
2            [summarize, visualize]
3   [prepare, summarize, visualize]
4    [access, construct, summarize]
5       [classification, evaluate]
6       [classification, evaluate]
7           [regression, evaluate]
8           [clustering, evaluate]
9                [optimize, tools]
10                [compare, tools]
11                  [unstructured]
12           [unstructured, tools]
13                [tools, compare]
0             python
1            process
2             access
3          construct
4          summarize
5          visualize
6            prepare
7     classification
8         regression
9         clustering
10          evaluate
11          optimize
12           compare
13      unstructured
14          workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
0             python
1            process
2             access
3          construct
4          summarize
5          visualize
6            prepare
7     classification
8         regression
9         clustering
10          evaluate
11          optimize
12           compare
13      unstructured
14          workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
0             python
1            process
2             access
3          construct
4          summarize
5          visualize
6            prepare
7     classification
8         regression
9         clustering
10          evaluate
11          optimize
12           compare
13      unstructured
14          workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
```

In that excerpt df_display has a key that is defined to be true or false. The value for each item in the dictionary (separated by commas `,`) is a lambda fucntion, both of which take a parameter `d`, and one of which returns the first column `d[d[columns[0]]` and the other fow which returns the whole data frame `d`.

In thee loop, we set index into the dictionary with the key `n_cols > 3` with `df_display[n_cols>3]` sometimes it will be true and other times it will be false, which matches the two keys defined in the dictionary. Then the parameter it takes is `d`, which we pass the whole data frame `df` with the `(df)` at the end of the line.

> **Try it Yourself!**
>
> What does the _ do?

```
Try using that dictionary outside of the loop.  What is the value for a give key
if you print it out like `df_display[ a_key_value]`? What if you use `True` or
`False` directly? what if you try a number? What is the type of that? What if you
pass something that's not a DataFrame as the parameter?  Make notes about these outputs
```

5.8. Questions after class

> **Note**
>
> add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

5.9. More Practice

- What `type` is the shape of a `pandas.DataFrame`?
- use a list comprehension to create a list that you could use as column names for data that consists of `N` measurements. Set `N=5` for now, but you suspect that the number might change.

- create a list of items with different types, then Create a dictionary with the types as keys using a dictionary comprehension. Dictionary comprehensions are similar to list comprehensions, in their form.

- Create a `lambda` function to print return the first 2 rows of a data frame

> **Ram Token Opportunity**
>
> Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

6. Exploratory Data Analysis

```
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. Data Frame from lists of Lists

On Friday, we collectively made a list of strings

```
# %load http://drsmb.co/310read

# share a sentence or a few words about how class is going or
# one takeaway you have from class so far.
#  and attribute with a name affter a hyphen(-)
# 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 = [
'Programming is a Practice. - Dr. Sarah Brown',
"So far it's going pretty good. - Matt",
'Pretty good pace, Github is still a little confusing is all - A',
"Class is going well, I'm really excited for the new skills that I will attain through
this course. - Diondra",
'Good straight forward - Adam',
'Good pace and engaging - Jacob',
'I like cheese - Anon',
'Going well, I enjoy python - Aiden',
"Class is going very well, I'm excited to learn more about manipulating data sets -
Greg",
'Really enjoying the class so far, Clear instructions and outcomes - Michael',
'So far this class is going well, very engaging lectures. - Anon',
'Great pace and notes have been really useful - Brandon',
'Good pace and easy to follow - Muhammad',
'Class is going well and engaging, but also a little difficult getting into the swing of
things - Isaiah',
'Well paced, informative, and helpful - Vinnie',
'Spectacular -Michael Jackson',
'Very interesting! I am enjoying it a lot so far. Getting to experience pandas as well
as using github/jupyter has been cool. One thing I would change though is slowing down
the pace a bit. - Max'
]
```

We can check that by using type, first on the whole thing

```
type(sentence_list)
```

```
list
```

And then we can index into the list. Lists can be indexed by integers:

```
sentence_list[4]
```

```
'Good straight forward - Adam'
```

is one item from the list and then check its type:

```
type(sentence_list[4])
```

```
str
```

First, we'll convert our list of strings, to a list of lists with a list comprehension. This will iterate over each string in that list and apply the string method `split`. Since we passed the parameter `'-'`, it will split at that character.

```
[sent_attr.split('-') for sent_attr in sentence_list]
```

Question From Class

The split method will split at every occurence of the character passed.

Try it Yourself

Try splitting based on a different character (eg `' '`). What happens?

```
[['Programming is a Practice. ', ' Dr. Sarah Brown'],
 ["So far it's going pretty good. ", ' Matt'],
 ['Pretty good pace, Github is still a little confusing is all ', ' A'],
 ["Class is going well, I'm really excited for the new skills that I will attain
through this course. ",
  ' Diondra'],
 ['Good straight forward ', ' Adam'],
 ['Good pace and engaging ', ' Jacob'],
 ['I like cheese ', ' Anon'],
 ['Going well, I enjoy python ', ' Aiden'],
 ["Class is going very well, I'm excited to learn more about manipulating data sets ",
  ' Greg'],
 ['Really enjoying the class so far, Clear instructions and outcomes ',
  ' Michael'],
 ['So far this class is going well, very engaging lectures. ', ' Anon'],
 ['Great pace and notes have been really useful ', ' Brandon'],
 ['Good pace and easy to follow ', ' Muhammad'],
 ['Class is going well and engaging, but also a little difficult getting into the swing
of things ',
  ' Isaiah'],
 ['Well paced, informative, and helpful ', ' Vinnie'],
 ['Spectacular ', 'Michael Jackson'],
 ['Very interesting! I am enjoying it a lot so far. Getting to experience pandas as
well as using github/jupyter has been cool. One thing I would change though is slowing
down the pace a bit. ',
  ' Max']]
```

If we save it to a variable, we can analyze it better, for example indexing it

```
list_of_lists = [sent_attr.split('-') for sent_attr in sentence_list]
list_of_lists[0]
```

```
['Programming is a Practice. ', ' Dr. Sarah Brown']
```

This is a list, which we can check with:

```
type(list_of_lists[0])
```

```
list
```

If we take one item from that, it's a string

```
list_of_lists[0][1]
```

```
' Dr. Sarah Brown'
```

The list of lists is the same length as our original sentence_list, because it was made from that.

```
len(sentence_list)
```

```
17
```

```
len(list_of_lists)
```

```
17
```

Then we can pass our list of lists to the DataFrame constructor and set the column headings with the `columns` parameter, which accepts a list.

```
pd.DataFrame([sent_attr.split('-') for sent_attr in sentence_list],
             columns=['sentence','attribution'])
```

Question From Class

While the list comprehension iterate over each string in that list, because it's a list, in order to index into it, we have to use an integer.

Iterating is taking each member in an object in turn, indexing is picking out a specified item. Iterating typically is done with the `for` keyword (either in a loop or a comprehension), indexing is done with `[]`

	sentence	attribution
0	Programming is a Practice.	Dr. Sarah Brown
1	So far it's going pretty good.	Matt
2	Pretty good pace, Github is still a little con...	A
3	Class is going well, I'm really excited for th...	Diondra
4	Good straight forward	Adam
5	Good pace and engaging	Jacob
6	I like cheese	Anon
7	Going well, I enjoy python	Aiden
8	Class is going very well, I'm excited to learn...	Greg
9	Really enjoying the class so far, Clear instru...	Michael
10	So far this class is going well, very engaging...	Anon
11	Great pace and notes have been really useful	Brandon
12	Good pace and easy to follow	Muhammad
13	Class is going well and engaging, but also a l...	Isaiah
14	Well paced, informative, and helpful	Vinnie
15	Spectacular	Michael Jackson
16	Very interesting! I am enjoying it a lot so fa...	Max

Hint

We built the list of lists here with a list comprehension because it's only a few items, but for a list of longer lists, you might use a for loop and `append`

6.3. Summarizing Data

```
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)
```

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()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mi
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	anko coffe produce
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethurama estat
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	Na
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugac
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuk developme tru

5 rows × 44 columns

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()
```

	Unnamed: 0	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000
mean	14.500000	168.000000	2013.964286	7.702500	7.630714	7.559643
std	8.225975	143.226317	1.346660	0.296156	0.303656	0.342469
min	1.000000	1.000000	2012.000000	6.750000	6.670000	6.500000
25%	7.750000	1.000000	2013.000000	7.580000	7.560000	7.397500
50%	14.500000	170.000000	2014.000000	7.670000	7.710000	7.670000
75%	21.250000	320.000000	2015.000000	7.920000	7.830000	7.770000
max	28.000000	320.000000	2017.000000	8.330000	8.080000	7.920000

8 rows × 21 columns

We can also select one variable at a time

```
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
balance_sorted[6]
```

```
7.5
```

We can also extract each of the statistics that the `describe` method calculates individually, by name. The quantiles are tricky, we can't ust `.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(.25)
```

```
7.5600000000000005
```

We can also pass other values

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

```
7.83
```

Calculate the mean of the `Aftertaste` column:

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

```
7.559642857142856
```

further reading

On the documentation page for describe the "ℹ 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.

6.4. What about the nonnumerical variables?

For example the color

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

```
0    Green
1      NaN
2    Green
3    Green
4    Green
Name: Color, 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?

What country is most prevalent in this dataset?

```
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'
```

Question From Class

Q: Can we calculate the mode to find the most prevalent? A: Yes. We can also use the mode function, which works on both numerical or nonnumerical values.

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

```
0    India
dtype: object
```

6.5. Questions After Class

6.5.1. General Questions

6.5.1.1. How to know what functions are compatible with other functions?

6.5.1.2. Best place to find all the individual functions based on the .describe() function

6.5.1.3. Are there panda functions to read files other than CSV?

6.5.2. Clarifying

6.5.2.1. Is sent_attr in the DataFrame loop its own variable that we assign, or is it a base Python function?

6.5.2.2. Difference between %load and how we've been importing links to datasets in previous classes?

6.5.2.3. When I ran [sent_attr.split('-') for sent_attr in sentence_list] in Jupyter, I got a nameerror

Why do we use the value quantile instead of using quartile since we can use mean to find mean?

6.5.3. Coure Admin and Assignment Questions

6.5.3.1. when are the achievements we earn in class going to be inputted in brightspace?

6.5.3.2. Is there anything we have to do for the assignment after commiting the changes to

the files?

6.5.3.3. When we push the homework are we pushing it to the portfolio?

6.5.3.4. Still a little unsure about what the num_numerical specifically is, is it just the number of numerical columns?

6.6. More Practice

1. Produce a table with the mean for each score.
2. Which variables have the most missing data?
3. What's the total number of bags of coffee produced?
4. Which ratings have similar ranges? (max, min)
5. Which rating are most consistent across coffees?
6. What score cutoff could you apply in order to select the top 10 best for Balance?

7. Visualization

```
import pandas as pd
import seaborn as sns
```

7.1. Jupyter FAQ

> **Question from class**
>
> Why doesn't my jupyter print things out that are on the last line?

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

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

```
'sarah'
```

How it diplsays 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, juypter uses that, and uses html to render the object in a more visually appealing way.

7.2. Revewi of describe

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'

coffee_df = pd.read_csv(arabica_data_url)
```

We can describe it again, to see it has mostly the same variables we saw before, but some different as well.

```
coffee_df.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000
mean	656.000763	153.887872	7.563806	7.518070	7.397696	7.533112
std	378.598733	129.733734	0.378666	0.399979	0.405119	0.381599
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	328.500000	14.500000	7.420000	7.330000	7.250000	7.330000
50%	656.000000	175.000000	7.580000	7.580000	7.420000	7.500000
75%	983.500000	275.000000	7.750000	7.750000	7.580000	7.750000
max	1312.000000	1062.000000	8.750000	8.830000	8.670000	8.750000

> **Question from class**
>
> Why do we need the `()` on the describe but not on just the data

As is often the case, again this comes back to the type.

```
type(coffee_df)
```

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

is a data frame which has the `_repr_html_` method

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	meta
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	meta
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wol
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	meta
...	...	...	...	...	...	...	
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	espera muni juch de fe
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	kope ekse b
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	memb bene atl con
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	bene se
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigral de

1311 rows × 44 columns

so it prints nicely as tdid the `coffee_df.describe()`

If we leave the `()` off we don't get nice formatting

```
coffee_df.describe
```

Note

notice that this one has the … in the columns and rows directions.

Try it Yourself

what can you check in a dataFrame to predict if it will display with or without the …?

Hint

when objects do not have the `_repr_html_` method their output includes the type

```
<bound method NDFrame.describe of       Unnamed: 0  Species                      Owner
Country.of.Origin  \
0              1  Arabica                  metad plc          Ethiopia
1              2  Arabica                  metad plc          Ethiopia
2              3  Arabica    grounds for health admin         Guatemala
3              4  Arabica         yidnekachew dabessa          Ethiopia
4              5  Arabica                  metad plc          Ethiopia
```

```
...          ...      ...                          ...              ...
1306        1307  Arabica    juan carlos garcia lopez            Mexico
1307        1308  Arabica     myriam kaplan-pasternak             Haiti
1308        1309  Arabica  exportadora atlantic, s.a.         Nicaragua
1309        1310  Arabica   juan luis alvarado romero         Guatemala
1310        1312  Arabica             bismarck castro          Honduras

                                  Farm.Name                  Lot.Number  \
0                                 metad plc                         NaN
1                                 metad plc                         NaN
2     san marcos barrancas "san cristobal cuch                      NaN
3        yidnekachew dabessa coffee plantation                      NaN
4                                 metad plc                         NaN
...                                     ...                       ...
1306                          el centenario                         NaN
1307                              200 farms                         NaN
1308                         finca las marías  017-053-0211/ 017-053-0212
1309                           finca el limon                       NaN
1310                             los hicaques                       103

                                          Mill  \
0                                    metad plc
1                                    metad plc
2                                          NaN
3                                    wolensu
4                                    metad plc
...                                        ...
1306  la esperanza, municipio juchique de ferrer, ve...
1307       coeb koperativ ekselsyo basen (350 members)
1308                        beneficio atlantic condega
1309                                beneficio serben
1310                               cigrah s.a de c.v.

                     ICO.Number                              Company  \
0                     2014/2015      metad agricultural developmet plc
1                     2014/2015      metad agricultural developmet plc
2                           NaN                                   NaN
3                           NaN  yidnekachew debessa coffee plantation
4                     2014/2015      metad agricultural developmet plc
...                         ...                                   ...
1306                1104328663                             terra mia
1307                       NaN                         haiti coffee
1308  017-053-0211/ 017-053-0212          exportadora atlantic s.a
1309                11/853/165                               unicafe
1310                13-111-053                cigrah s.a de c.v

          Altitude  ...       Color Category.Two.Defects  \
0         1950-2200  ...       Green                   0
1         1950-2200  ...       Green                   1
2     1600 - 1800 m  ...         NaN                   0
3         1800-2200  ...       Green                   2
4         1950-2200  ...       Green                   2
...             ...  ...         ...                 ...
1306            900  ...        None                  20
1307         ~350m  ...  Blue-Green                   16
1308           1100  ...       Green                   5
1309           4650  ...       Green                   4
1310           1400  ...       Green                   2

               Expiration                Certification.Body  \
0          April 3rd, 2016  METAD Agricultural Development plc
1          April 3rd, 2016  METAD Agricultural Development plc
2           May 31st, 2011        Specialty Coffee Association
3         March 25th, 2016  METAD Agricultural Development plc
4          April 3rd, 2016  METAD Agricultural Development plc
...                    ...                                 ...
1306  September 17th, 2013                            AMECAFE
1307        May 24th, 2013        Specialty Coffee Association
1308        June 6th, 2018        Instituto Hondureño del Café
1309        May 24th, 2013        Asociacion Nacional Del Café
1310      April 28th, 2018        Instituto Hondureño del Café

                       Certification.Address  \
0     309fcf77415a3661ae83e027f7e5f05dad786e44
1     309fcf77415a3661ae83e027f7e5f05dad786e44
2     36d0d00a3724338ba7937c52a378d085f2172daa
3     309fcf77415a3661ae83e027f7e5f05dad786e44
4     309fcf77415a3661ae83e027f7e5f05dad786e44
...                                        ...
1306  59e396ad6e22a1c22b248f958e1da2bd8af85272
1307  36d0d00a3724338ba7937c52a378d085f2172daa
1308  b4660a57e9f8cc613ae5b8f02bfce8634c763ab4
1309  b1f20fe3a819fd6b2ee0eb8fdc3da256604f1e53
1310  b4660a57e9f8cc613ae5b8f02bfce8634c763ab4

                       Certification.Contact unit_of_measurement  \
0     19fef5a731de2db57d16da10287413f5f99bc2dd                   m
```

```
1     19fef5a731de2db57d16da10287413f5f99bc2dd                  m
2     0878a7d4b9d35ddbf0fe2ce69a2062cceb45a660                  m
3     19fef5a731de2db57d16da10287413f5f99bc2dd                  m
4     19fef5a731de2db57d16da10287413f5f99bc2dd                  m
...                                        ...                ...
1306  0eb4ee5b3f47b20b049548a2fd1e7d4a2b70d0a7                  m
1307  0878a7d4b9d35ddbf0fe2ce69a2062cceb45a660                  m
1308  7f521ca403540f81ec99daec7da19c2788393880                  m
1309  724f04ad10ed31dbb9d260f0dfd221ba48be8a95                 ft
1310  7f521ca403540f81ec99daec7da19c2788393880                  m

     altitude_low_meters altitude_high_meters altitude_mean_meters
0                1950.00              2200.00               2075.00
1                1950.00              2200.00               2075.00
2                1600.00              1800.00               1700.00
3                1800.00              2200.00               2000.00
4                1950.00              2200.00               2075.00
...                  ...                  ...                   ...
1306              900.00               900.00                900.00
1307              350.00               350.00                350.00
1308             1100.00              1100.00               1100.00
1309             1417.32              1417.32               1417.32
1310             1400.00              1400.00               1400.00

[1311 rows x 44 columns]>
```

so lets check the type of that.

```
type(coffee_df.describe)
```

```
method
```

it's a `bound method` or a function that *will* be applied to the DataFrame, but we didn't actually run the method. To see that it hasn't run, we can use an ipython[1] magic `%%timeit`

```
%%timeit
coffee_df.describe
```

```
94.7 ns ± 0.0305 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)
```

```
%%timeit
coffee_df.describe()
```

```
39.4 ms ± 98.7 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Note that without the `()` it runs much much faster, signaling that it did less finding the method, is less calcuation than computing statistics on the data

Further reading

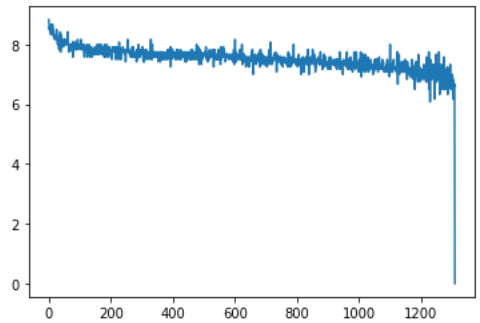
The magic functions can be defined for a cell or a line. Timeit is actually a standard python library that you can call on the command line or in a python script as well, but juypter, by way of ipython gives us easy access to it with nice defaults.

7.3. Basic plots in pandas

Pandas gives us basic plots.

```
coffee_df['Flavor'].plot()
```

```
<AxesSubplot:>
```



Since we chose a series, it plotted that data as line vs the index.

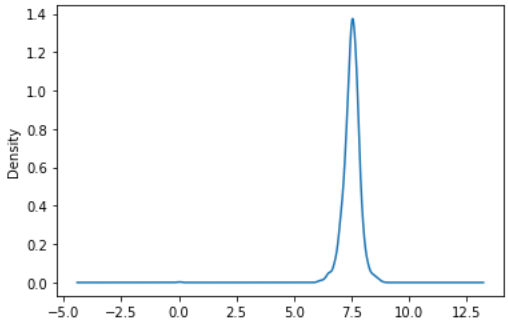
```
coffee_df.index
```

```
RangeIndex(start=0, stop=1311, step=1)
```

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.

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

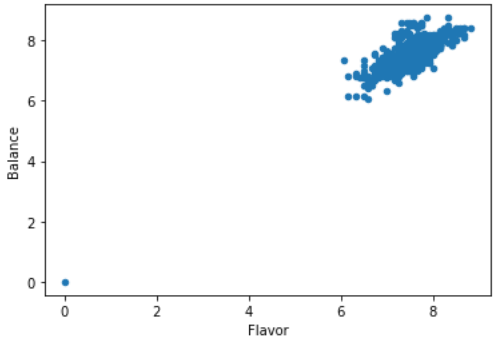
```
<AxesSubplot:ylabel='Density'>
```



We can also plot two variables as a scatter plot, by specifying the `x`, `y` and `kind`

```
coffee_df.plot(x='Flavor',y='Balance', kind='scatter')
```

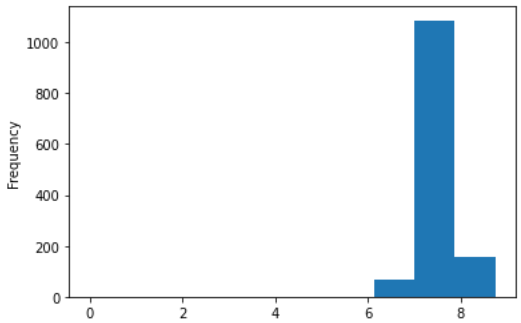
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



Let's Make a histogram plot of the Balance variable

```
coffee_df['Balance'].plot(kind='hist')
```

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



Question from class

Can we plot two histograms with `coffee_df['Balance']['Flavor'].plot(kind='hist')`

```
:tags: ["raises-exception"]

coffee_df['Balance']['Flavor'].plot(kind='hist')
```

```
  File "/tmp/ipykernel_1743/3818478052.py", line 1
    :tags: ["raises-exception"]
    ^
SyntaxError: invalid syntax
```

Let's break down why that errors. When we append things to the left, python interprets them by passing the output of one step to the input of the next one.
So `coffee_df['Balance'].plot(kind='hist')` first made a series, then plotted it. In the above, we again got the series, which works

```
coffee_df['Balance'].head(2)
```

```
0    8.42
1    8.42
Name: Balance, dtype: float64
```

> **Note**
>
> Since I know these will be DataFrames, I'm appending the `head()` method to reduce the output for presenation.

But then, we tried to index it with 'Flavor', but we don't have that any more

```
coffee_df['Balance']['Flavor']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1743/1214282069.py in <module>
----> 1 coffee_df['Balance']['Flavor']

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/series.py in __getitem__(self, key)
    940
    941         elif key_is_scalar:
--> 942             return self._get_value(key)
    943
    944         if is_hashable(key):

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/series.py in _get_value(self, label, takeable)
   1049
   1050         # Similar to Index.get_value, but we do not fall back to positional
-> 1051         loc = self.index.get_loc(label)
   1052         return self.index._get_values_for_loc(self, loc, label)
   1053

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
    386                 except ValueError as err:
    387                     raise KeyError(key) from err
--> 388             raise KeyError(key)
    389         return super().get_loc(key, method=method, tolerance=tolerance)
    390

KeyError: 'Flavor'
```

So we get a key error and we know this is the part of the line we have to change.

We need to index into the DataFrame and pick two columns at once. When we index, we can use the name of a variable as a string or a list. We can buil this list on the fly and python exectues fromt he inside out.
The outer `[ ]` index and the inner `[ ]` make alist

```
coffee_df[['Balance','Flavor']].head(2)
```

	Balance	Flavor
0	8.42	8.83
1	8.42	8.67

we could also build the list first, then index for readability

```
hist_vars = ['Balance','Flavor'].head(2)
coffee_df[hist_vars]
```

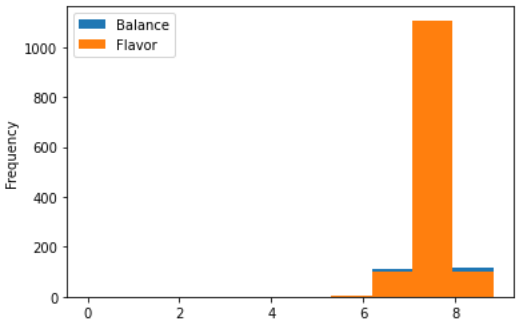
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1743/2943951791.py in <module>
----> 1 hist_vars = ['Balance','Flavor'].head(2)
      2 coffee_df[hist_vars]

AttributeError: 'list' object has no attribute 'head'
```

This gives us a data frame, which we can plot.

```
coffee_df[['Balance','Flavor']].plot(kind='hist')
```

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



We'll see ways to improve this on Friday.

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.

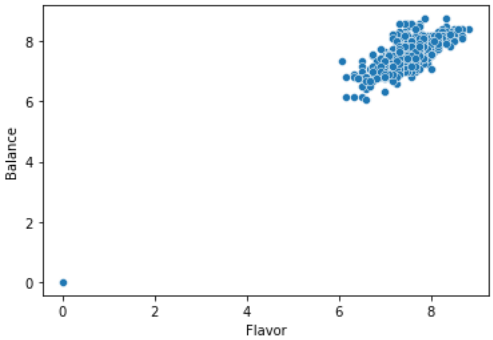
Pandas and seaborn use matplotlib under the hood.

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.

We can get that basic plot back.

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

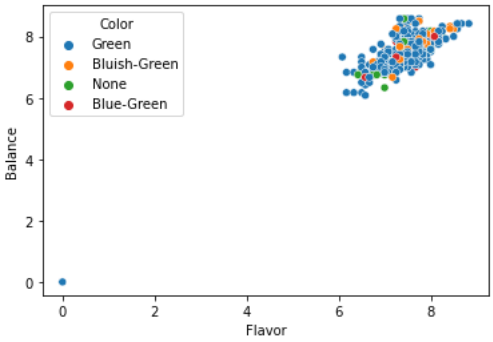
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



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

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

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

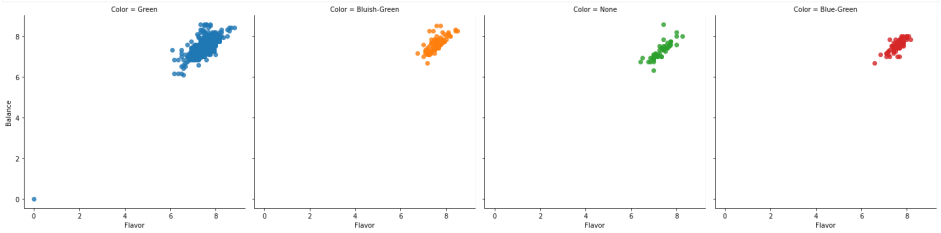


From this we can see that the color doesn't appear to be related to the flavor or balance scores, but that the flavor and balacne are related.

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. We'll turn that off for now, with `,fit_reg=False`.

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
               col='Color',fit_reg=False)
```

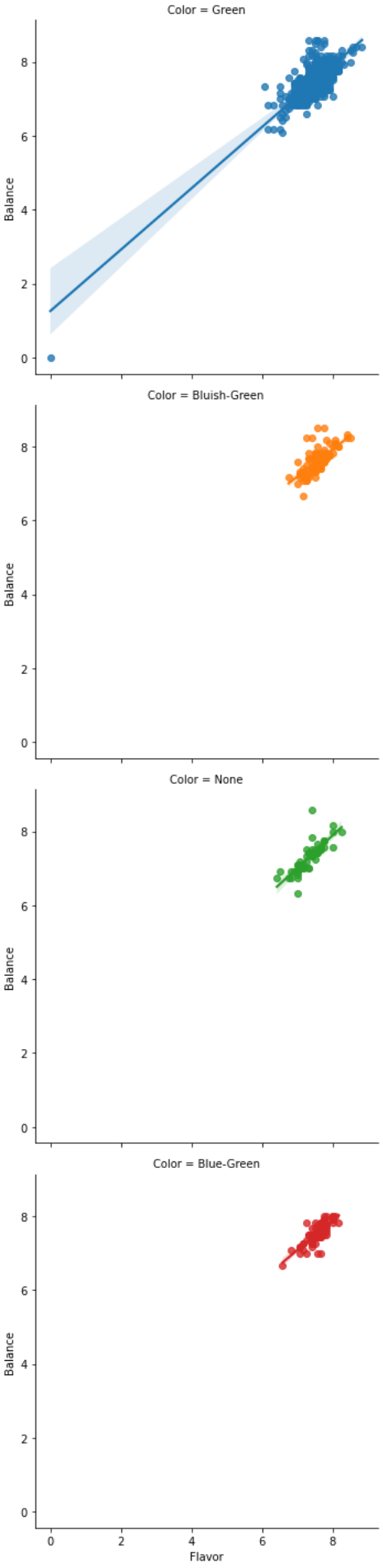
```
<seaborn.axisgrid.FacetGrid at 0x7f60506d8c50>
```



`col` stands for column. We can also use `row`

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
               row='Color')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6050300b50>
```



We can also use both together:

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
               row='Color',col='Country.of.Origin')
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipykernel_1743/4097410330.py in <module>
      1 sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
----> 2                row='Color',col='Country.of.Origin')

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/seaborn/_decorators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/seaborn/regression.py in lmplot(x, y, data, hue, col, row, palette, col_wrap,
height, aspect, markers, sharex, sharey, hue_order, col_order, row_order, legend,
legend_out, x_estimator, x_bins, x_ci, scatter, fit_reg, ci, n_boot, units, seed,
order, logistic, lowess, robust, logx, x_partial, y_partial, truncate, x_jitter,
y_jitter, scatter_kws, line_kws, facet_kws, size)
    643         scatter_kws=scatter_kws, line_kws=line_kws,
    644     )
--> 645     facets.map_dataframe(regplot, x=x, y=y, **regplot_kws)
    646     facets.set_axis_labels(x, y)
    647

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/seaborn/axisgrid.py
in map_dataframe(self, func, *args, **kwargs)
    782         for i, val in enumerate(args[:2]):
    783             axis_labels[i] = val
--> 784         self._finalize_grid(axis_labels)
    785
    786         return self

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/seaborn/axisgrid.py
in _finalize_grid(self, axlabels)
    813         self.set_axis_labels(*axlabels)
    814         self.set_titles()
--> 815         self.tight_layout()
    816
    817     def facet_axis(self, row_i, col_j, modify_state=True):

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/seaborn/axisgrid.py
in tight_layout(self, *args, **kwargs)
     86         if self._tight_layout_pad is not None:
     87             kwargs.setdefault("pad", self._tight_layout_pad)
---> 88         self._figure.tight_layout(*args, **kwargs)
     89
     90     def add_legend(self, legend_data=None, title=None, label_order=None,

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/figure.py
in tight_layout(self, pad, h_pad, w_pad, rect)
   3164             kwargs = get_tight_layout_figure(
   3165                 self, self.axes, subplotspec_list, renderer,
-> 3166                 pad=pad, h_pad=h_pad, w_pad=w_pad, rect=rect)
   3167         if kwargs:
   3168             self.subplots_adjust(**kwargs)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/tight_layout.py in get_tight_layout_figure(fig, axes_list,
subplotspec_list, renderer, pad, h_pad, w_pad, rect)
    315                                     subplot_list=subplot_list,
    316                                     ax_bbox_list=ax_bbox_list,
--> 317                                     pad=pad, h_pad=h_pad, w_pad=w_pad)
    318
    319     # kwargs can be none if tight_layout fails...

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/tight_layout.py in auto_adjust_subplotpars(fig, renderer,
nrows_ncols, num1num2_list, subplot_list, ax_bbox_list, pad, h_pad, w_pad, rect)
     82             if ax.get_visible():
     83                 try:
---> 84                     bb += [ax.get_tightbbox(renderer, for_layout_only=True)]
     85                 except TypeError:
     86                     bb += [ax.get_tightbbox(renderer)]

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/axes/_base.py in get_tightbbox(self, renderer, call_axes_locator,
bbox_extra_artists, for_layout_only)
   4444                 if bb_yaxis:
```

```
   4445                     bb.append(bb_yaxis)
-> 4446         self._update_title_position(renderer)
   4447         axbbox = self.get_window_extent(renderer)
   4448         bb.append(axbbox)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/axes/_base.py in _update_title_position(self, renderer)
   2810                 if (ax.xaxis.get_ticks_position() in ['top', 'unknown']
   2811                         or ax.xaxis.get_label_position() == 'top'):
-> 2812                     bb = ax.xaxis.get_tightbbox(renderer)
   2813                 else:
   2814                     bb = ax.get_window_extent(renderer)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/axis.py
in get_tightbbox(self, renderer, for_layout_only)
   1084             return
   1085
-> 1086         ticks_to_draw = self._update_ticks()
   1087
   1088         self._update_label_position(renderer)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/axis.py
in _update_ticks(self)
   1027         """
   1028         major_locs = self.get_majorticklocs()
-> 1029         major_labels = self.major.formatter.format_ticks(major_locs)
   1030         major_ticks = self.get_major_ticks(len(major_locs))
   1031         self.major.formatter.set_locs(major_locs)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/ticker.py
in format_ticks(self, values)
    260     def format_ticks(self, values):
    261         """Return the tick labels for all the ticks at once."""
--> 262         self.set_locs(values)
    263         return [self(value, i) for i, value in enumerate(values)]
    264

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/ticker.py
in set_locs(self, locs)
    804                 self._compute_offset()
    805             self._set_order_of_magnitude()
--> 806             self._set_format()
    807
    808     def _compute_offset(self):

KeyboardInterrupt:
```

How could we choose which countries to select to make this not show the ones with very few points?

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

Or we can focus on the countried, but wrap them.

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
               col='Country.of.Origin',col_wrap=5)
```

7.5. Questions after class

> **Ram Token Opportunity**
>
> add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

7.6. More practice

1. Plot the kde for the `Aftertaste`
2. How does `Total.Cup.Points` vary by `Certification.Body`
3. Are moisture and sweetness related? Does that relationship vary by Color?

---

[1] the kernel of python we're using

8. Exploratory Data Analysis

```
import pandas as pd
import seaborn as sns
```

```
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)
```

Which of the following scores is distributed most similarly to Sweetness?

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

First step is to subset the data:

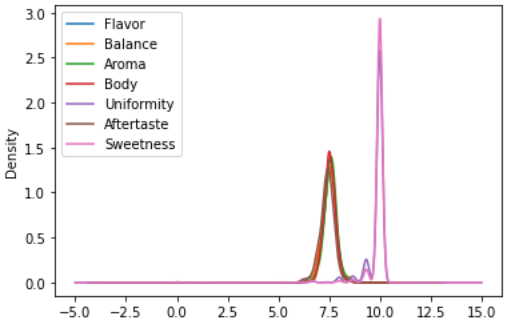
```
coffee_df[scores_of_interest].head(2)
```

	Flavor	Balance	Aroma	Body	Uniformity	Aftertaste	Sweetness
0	8.83	8.42	8.67	8.50	10.0	8.67	10.0
1	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')
```

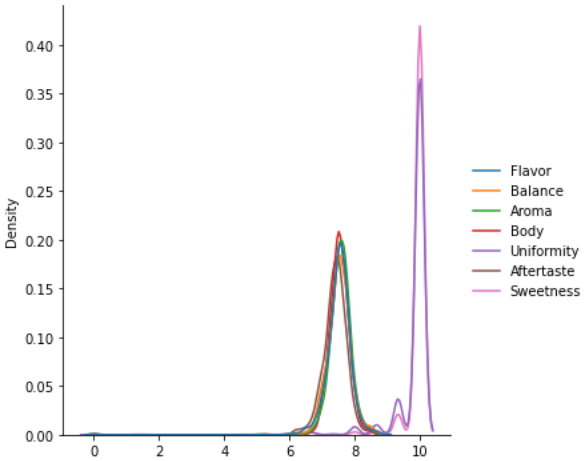
```
<AxesSubplot:ylabel='Density'>
```



We could also do it with seaborn

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

```
<seaborn.axisgrid.FacetGrid at 0x7f4b682a10d0>
```



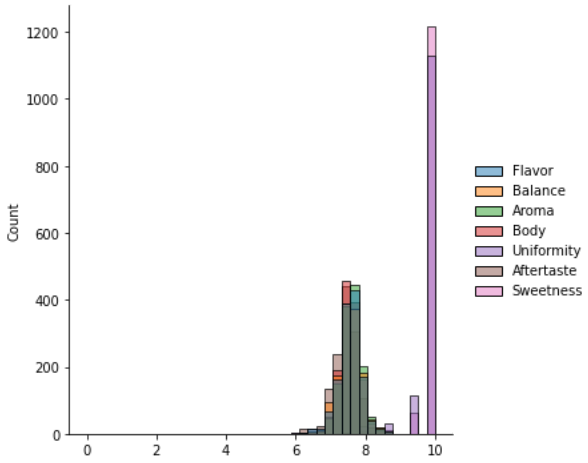
If we forget the parameter `kind`, we get its default value, which is histogram

```
``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)
```

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

```
<seaborn.axisgrid.FacetGrid at 0x7f4b681fead0>
```



8.1. Summarizing with two 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 `value_counts()` on the `Country.of.Origin` column.

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

```
Mexico      236
Colombia    183
Name: Country.of.Origin, dtype: int64
```

The data also has `Number.of.Bags` however. How can we check which has the most bags?

We can do this with groupby.

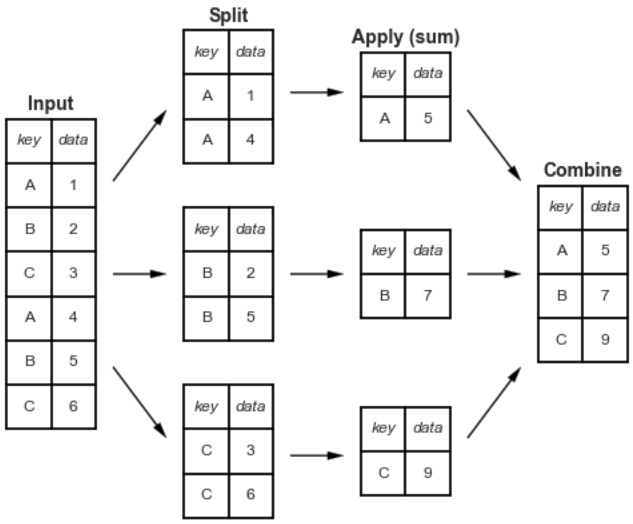
```
coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

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.

```
Country.of.Origin
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
Name: Number.of.Bags, dtype: int64
```

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. How doe Groupby Work?

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 0x7f4b67d09b50>
```

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

```python
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'])
```

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.

	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

> **Question from class**
>
> How can we sort it?

First, we'll make it a variable to keep the code legible, then we'll use `sort_values()`

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

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

Which, by default uses ascending order, the method has an `ascending` parameter and its default value is `True`, so we can switch it to `False` to get descending

```
bag_total_df.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
```

8.3. Customizing Data Summaries

We've looked at an overall summary with `describe` on all the variables, describe on one variable, individual statistics on all variables, and individual statistics on one variable. We can also build summaries of multiple varialbes with a custom subset of summary statistics, with `aggregate` or using its alias `agg`

```
country_grouped.agg({'Number.of.Bags':'sum',
                     'Balance':['mean','count'],})
```

	Number.of.Bags	Balance	
	sum	mean	count
Country.of.Origin			
Brazil	30534	7.531515	132
Burundi	520	7.415000	2
China	55	7.548125	16
Colombia	41204	7.708415	183
Costa Rica	10354	7.637255	51
Cote d?Ivoire	2	7.080000	1
Ecuador	1	7.830000	1
El Salvador	4449	7.711429	21
Ethiopia	11761	7.972273	44
Guatemala	36868	7.469890	181
Haiti	390	7.056667	6
Honduras	13167	7.163962	53
India	20	7.420000	1
Indonesia	1658	7.520000	20
Japan	20	7.830000	1
Kenya	3971	7.800400	25
Laos	81	7.416667	3
Malawi	557	7.371818	11
Mauritius	1	7.170000	1
Mexico	24140	7.328686	236
Myanmar	10	7.133750	8
Nicaragua	6406	7.278462	26
Panama	537	7.875000	4
Papua New Guinea	7	8.250000	1
Peru	2336	7.666000	10
Philippines	259	7.400000	5
Rwanda	150	7.750000	1
Taiwan	1914	7.426000	75
Tanzania, United Republic Of	3760	7.469750	40
Thailand	1310	7.524063	32
Uganda	3868	7.660769	26
United States	361	7.947500	8
United States (Hawaii)	833	7.644110	73
United States (Puerto Rico)	71	7.647500	4
Vietnam	10	7.547143	7
Zambia	13	7.420000	1

🧠 Learning Tip

When a person reads a line of code, they have to use their working memory to hold the whole thing in order to make sense of it. Human working memory only holds 5-9 things at a time, that's why a phone number is 7 digits without the area code, (when people used to have to actually dial the digits, they also didn't need the area codes for short-distance calls, which were most of the calls).

How many concepts does a person have to hold in working memory at once to parse the second version?

If you are writing the code, and building it up, you hold the previous pieces in your memory differently than a person coming to it for the first time.

We could also string this together, but splitting into interim variables makes code more readable. Shorter lines are easier to read (and sometimes auto-enforced on projects). Also, by giving a good variable name to the interim states we get more description, which can help a human better read it.

```
coffee_df.groupby('Country.of.Origin').agg({'Number.of.Bags':'sum','Balance':
['mean','count'],})
```

	Number.of.Bags	Balance	
	sum	mean	count
Country.of.Origin			
Brazil	30534	7.531515	132
Burundi	520	7.415000	2
China	55	7.548125	16
Colombia	41204	7.708415	183
Costa Rica	10354	7.637255	51
Cote d?Ivoire	2	7.080000	1
Ecuador	1	7.830000	1
El Salvador	4449	7.711429	21
Ethiopia	11761	7.972273	44
Guatemala	36868	7.469890	181
Haiti	390	7.056667	6
Honduras	13167	7.163962	53
India	20	7.420000	1
Indonesia	1658	7.520000	20
Japan	20	7.830000	1
Kenya	3971	7.800400	25
Laos	81	7.416667	3
Malawi	557	7.371818	11
Mauritius	1	7.170000	1
Mexico	24140	7.328686	236
Myanmar	10	7.133750	8
Nicaragua	6406	7.278462	26
Panama	537	7.875000	4
Papua New Guinea	7	8.250000	1
Peru	2336	7.666000	10
Philippines	259	7.400000	5
Rwanda	150	7.750000	1
Taiwan	1914	7.426000	75
Tanzania, United Republic Of	3760	7.469750	40
Thailand	1310	7.524063	32
Uganda	3868	7.660769	26
United States	361	7.947500	8
United States (Hawaii)	833	7.644110	73
United States (Puerto Rico)	71	7.647500	4
Vietnam	10	7.547143	7
Zambia	13	7.420000	1

Question from Class

What does the `count` do? or how can we figure it out?

In this case, let's compare `count` to `shape` we know that shape returns the number of rows and columns. Also `shape` is an attribute, not a method (so no `()` for shape). However, there's a more important difference.

```
coffee_df['Balance','Farm.Name','Lot.Number'].shape
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.12/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/3.7.12/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.12/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: ('Balance', 'Farm.Name', 'Lot.Number')

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

KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1796/4170084615.py in <module>
----> 1 coffee_df['Balance','Farm.Name','Lot.Number'].shape

/opt/hostedtoolcache/Python/3.7.12/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/3.7.12/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: ('Balance', 'Farm.Name', 'Lot.Number')
```

```
coffee_df['Balance','Farm.Name','Lot.Number'].count()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.12/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/3.7.12/x64/lib/python3.7/site-
packages/pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.12/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: ('Balance', 'Farm.Name', 'Lot.Number')

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

KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1796/48982150.py in <module>
----> 1 coffee_df['Balance','Farm.Name','Lot.Number'].count()

/opt/hostedtoolcache/Python/3.7.12/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/3.7.12/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: ('Balance', 'Farm.Name', 'Lot.Number')
```

Here we get different number for `Balance` and `Farm Name` The shape tells us how many rows there are, while count tells us how many are not null.

We can verify visually that some are null with:

```
coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc

5 rows × 44 columns

> ⚠ **Correction**
>
> This response is slightly corrected from what I said in class, because in the balance column, it does match, but in general it doesn't. `count` on grouped will only be the same as `value_count` when `count` is applied to a column that does not have any missing values where the grouping variable has a value.

On the balance column alone in class, we checked that the grouped count matched the country value counts.

```
country_grouped[['Balance','Farm.Name','Lot.Number']].count().sort_values(by='Balance'
    ascending=False)
```

```
  File "/tmp/ipykernel_1796/3556086896.py", line 2
    ascending=False)
              ^
SyntaxError: invalid syntax
```

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

```
Mexico                          236
Colombia                        183
Guatemala                       181
Brazil                          132
Taiwan                           75
United States (Hawaii)           73
Honduras                         53
Costa Rica                       51
Ethiopia                         44
Tanzania, United Republic Of     40
Thailand                         32
Uganda                           26
Nicaragua                        26
Kenya                            25
El Salvador                      21
Indonesia                        20
China                            16
Malawi                           11
Peru                             10
United States                     8
Myanmar                           8
Vietnam                           7
Haiti                             6
Philippines                       5
Panama                            4
United States (Puerto Rico)       4
Laos                              3
Burundi                           2
Ecuador                           1
Rwanda                            1
Japan                             1
Zambia                            1
Papua New Guinea                  1
Mauritius                         1
Cote d?Ivoire                     1
India                             1
Name: Country.of.Origin, dtype: int64
```

In class, they were the same, because `Balance` doesn't have missing values. `Farm.Name` and `Lot.Number` havea lot of missing values, so they're different numbers.

Another function we could use when we first examine a dataset is `info` this tells us about the NaN values up front.

```
coffee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1311 entries, 0 to 1310
Data columns (total 44 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             1311 non-null   int64
 1   Species                1311 non-null   object
 2   Owner                  1304 non-null   object
 3   Country.of.Origin      1310 non-null   object
 4   Farm.Name              955 non-null    object
 5   Lot.Number             270 non-null    object
 6   Mill                   1001 non-null   object
 7   ICO.Number             1165 non-null   object
 8   Company                1102 non-null   object
 9   Altitude               1088 non-null   object
 10  Region                 1254 non-null   object
 11  Producer               1081 non-null   object
 12  Number.of.Bags         1311 non-null   int64
 13  Bag.Weight             1311 non-null   object
 14  In.Country.Partner     1311 non-null   object
 15  Harvest.Year           1264 non-null   object
 16  Grading.Date           1311 non-null   object
 17  Owner.1                1304 non-null   object
 18  Variety                1110 non-null   object
 19  Processing.Method      1159 non-null   object
 20  Aroma                  1311 non-null   float64
 21  Flavor                 1311 non-null   float64
 22  Aftertaste             1311 non-null   float64
 23  Acidity                1311 non-null   float64
 24  Body                   1311 non-null   float64
 25  Balance                1311 non-null   float64
 26  Uniformity             1311 non-null   float64
 27  Clean.Cup              1311 non-null   float64
 28  Sweetness              1311 non-null   float64
 29  Cupper.Points          1311 non-null   float64
 30  Total.Cup.Points       1311 non-null   float64
 31  Moisture               1311 non-null   float64
 32  Category.One.Defects   1311 non-null   int64
 33  Quakers                1310 non-null   float64
 34  Color                  1095 non-null   object
 35  Category.Two.Defects   1311 non-null   int64
 36  Expiration             1311 non-null   object
 37  Certification.Body     1311 non-null   object
 38  Certification.Address  1311 non-null   object
 39  Certification.Contact  1311 non-null   object
 40  unit_of_measurement    1311 non-null   object
 41  altitude_low_meters    1084 non-null   float64
 42  altitude_high_meters   1084 non-null   float64
 43  altitude_mean_meters   1084 non-null   float64
dtypes: float64(16), int64(4), object(24)
memory usage: 450.8+ KB
```

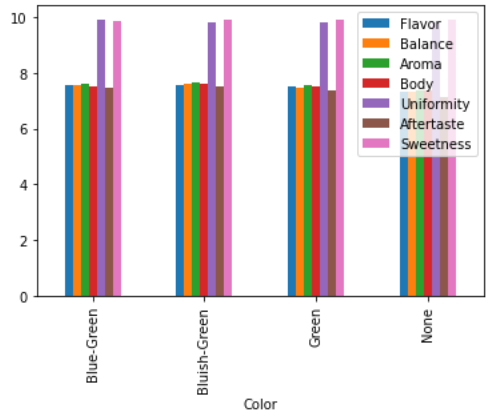
8.4. Using summaries for visualization

> **Important**
>
> This section is an extension, that we didn't get to in class, but might help in your assignment

For example, we can group by color and take the mean of each of the scores from the beginning of class and then use a bar chart.

```
color_grouped = coffee_df.groupby('Color')[scores_of_interest].mean()
color_grouped.plot(kind='bar')
```

```
<AxesSubplot:xlabel='Color'>
```



8.5. Questions after class

> **Ram Token Opportunity**
>
> add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

8.6. 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.

9. Reshaping Data

Today, we'll begin reshaping data. We'll cover:

- filtering
- applying a function to all rows
- what is tidy data
- reshaping data into tidy data

First some setup:

```
import pandas as pd
import seaborn as sns

sns.set_theme(font_scale=2)
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
```

9.1. Cleaning Data

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
- changing the representation of information

9.2. 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 + file,na_values= '') for file in datasets]
```

```
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/3.7.12/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/3.7.12/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/3.7.12/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/3.7.12/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/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3444: 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/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3444: 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.

9.3. 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

> ⚠ **Correction**
>
> I fixed the three data files so the spaces can be removed. You will need to

```
treat_df.melt(value_vars = ['treatmenta','treatmentb'],
              id_vars = ['name'],
              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

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

```
tidy_treat_df.groupby('name').mean()
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3444: 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)
```

name
Jane Doe
John Smith
Mary Johnson

9.4. Filtering Data by a column

Let's go back to the coffee dataset

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

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN
4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN
5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015

5 rows × 43 columns

Recall on Friday we computed the total number of bags per country.

```
# compute total bags per country
bag_total_df = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

We can subset this to get only the countries with over 15000 using a boolean mask:

```
bag_total_df[bag_total_df>15000]
```

```
Country.of.Origin
Brazil       30534
Colombia     41204
Guatemala    36868
Mexico       24140
Name: Number.of.Bags, dtype: int64
```

what we put in the `[]` has to be the same length and each element has to be boolean

```
len(bag_total_df>15000)
```

```
36
```

```
mask = bag_total_df>15000
type(mask[0])
```

```
numpy.bool_
```

9.5. Augmenting a dataset

We want the names of the countries as a list, so we extract the index of that series and then cast it to a list.

```
high_prod_countries =  list(bag_total_df[bag_total_df>15000].index)
```

Next we want to be able to check if a country is in this list, so we'll make a lambda that can do that

```
high_prod = lambda c: c in high_prod_countries
```

Recall, the `lambda` keyword makes a function

```
type(high_prod)
```

```
function
```

We can test it

```
high_prod('Mexico'), high_prod('Ethiopa')
```

```
(True, False)
```

Now, we can apply that lambda function to each country in our whole coffee data frame. and save that to a new DataFrame.

```
coffee_df['high_production'] = coffee_df['Country.of.Origin'].apply(high_prod)
```

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015
3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN
4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN
5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015

5 rows × 44 columns

Finally, we can filter the whole data frame using that new column.

```
high_prod_coffee_df = coffee_df[coffee_df['high_production']]
```

Question from class

How can we get the ones not on that list?

```
low_prod_coffee_df = coffee_df[coffee_df['high_production']==False]
```

Try it Yourself

Replace the FIXMEs in the excerpt below to reshape the data to have a value column with the value of the score and a Score column that indicates which score is in that . Keep the color and country as values

```
scores_of_interest = ['Balance','Aroma','Body','Aftertaste']
attrs_of_interest = ['Country.of.Origin','Color']
high_prod_coffee_df_melted = high_prod_coffee_df.melt(
        id_vars = FIXME,
        value_vars = FIXME,
        value_name = 'Value',
        var_name = 'Score')
```

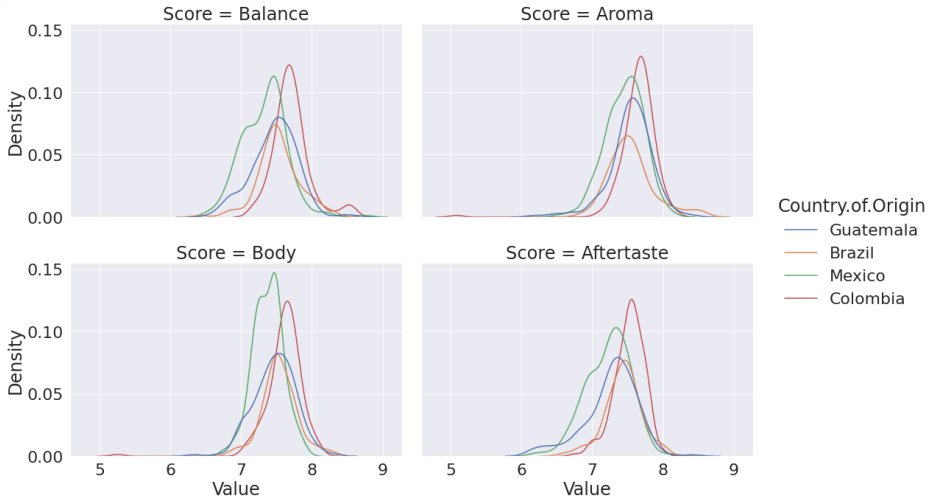
so that it looks like the following

	Country.of.Origin	Color	Score	Value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00

Try it Yourself

Plot the distribution of each score on a separate subplot and use a different color for each country. Use a kde for the distributions.

```
<seaborn.axisgrid.FacetGrid at 0x7f20df50c210>
```



10. More Reshaping

Continuing from Friday.

```
import pandas as pd
import seaborn as sns

# make plots look nicer and increase font size
sns.set_theme(font_scale=2)
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)

# compute ___ per ___
bag_total_df = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()

# subset the summary Series for countries with over 15000 total and store as a list
high_prod_countries = list(bag_total_df[bag_total_df>15000].index)

# a lambda function that checks if a string c is one of the
#  countries in high_prod_countries
high_prod = lambda c: c in high_prod_countries

# add a column that indicates that the country is a high producer
coffee_df['high_production'] = coffee_df['Country.of.Origin'].apply(high_prod)



# filter based on production level threshold
high_prod_coffee_df = coffee_df[coffee_df['high_production']]
```

What happened when we filtered the data?

```
coffee_df.shape, high_prod_coffee_df.shape
```

```
((1311, 45), (732, 45))
```

We have many fewer rows.

Now that we've filtered the data. Let's practice reshaping data to by Tidy again.

```
# replace the FIXMEs
scores_of_interest = ['Balance','Aroma','Body','Aftertaste']
attrs_of_interest = ['Country.of.Origin','Color']
high_prod_coffee_df_melted = high_prod_coffee_df.melt(
    id_vars = attrs_of_interest,
    value_vars = scores_of_interest,
    var_name = 'Score')
```

What happened?

```
high_prod_coffee_df_melted.shape
```

```
(2928, 4)
```

Now the shape is 4 times as long (because the length of the list we passed to value_vars is 4). And it has 4 columns: the length of the list we passed to `id_vars` + 2 (variable, value)

```
len(scores_of_interest)
```

```
4
```

```
len(scores_of_interest)*len(high_prod_coffee_df)
```

```
2928
```

We can seee the column names and what they have in them here:

```
high_prod_coffee_df_melted.head()
```

	Country.of.Origin	Color	Score	value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00

Note that we passed a value to `var_name` to make that column named “Score”. We could also not pass that

```
high_prod_coffee_df.melt(
    id_vars = attrs_of_interest,
    value_vars = scores_of_interest)
```

	Country.of.Origin	Color	variable	value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00
...	...	...	...	...
2923	Mexico	Green	Aftertaste	6.42
2924	Mexico	Green	Aftertaste	6.83
2925	Brazil	Green	Aftertaste	6.83
2926	Mexico	None	Aftertaste	6.25
2927	Guatemala	Green	Aftertaste	6.67

2928 rows × 4 columns

then we have `variable` and `value` as column names.

> **</> Try it yourself**
>
> How could you rename the `value` column?

The head has only ‘Balance’ in the ‘Score’ column, we could use `sample` to pick a random subset of the rows instead to see different values.

```
high_prod_coffee_df_melted.sample(5)
```

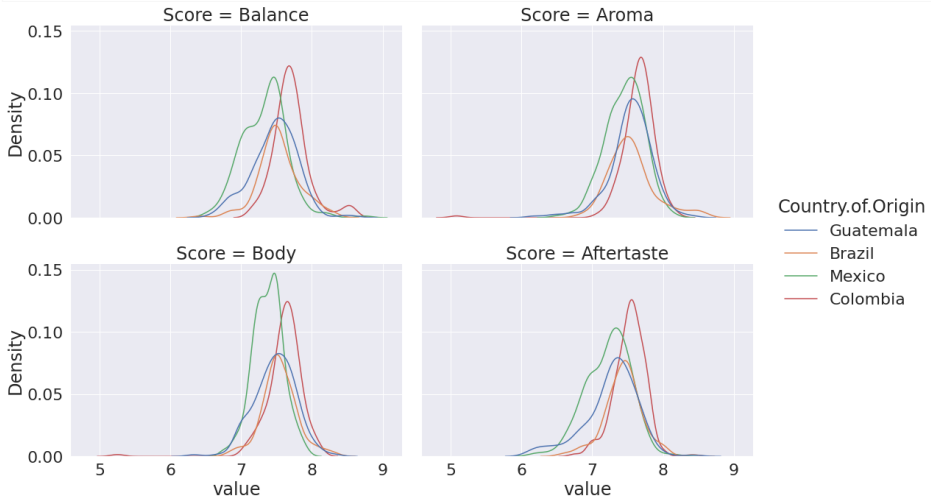
	Country.of.Origin	Color	Score	value
848	Mexico	Green	Aroma	7.75
2032	Mexico	Green	Body	7.17
1264	Colombia	Green	Aroma	7.50
4	Brazil	Green	Balance	8.00
2045	Guatemala	Green	Body	7.00

What does this let us do?

One thing is it makes plots easier, because seaborn is organized around tidy data.

```
sns.displot(data= high_prod_coffee_df_melted,
            x='value',hue='Country.of.Origin',
            col = 'Score', col_wrap=2, kind='kde',aspect =1.5)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb7b074d210>
```



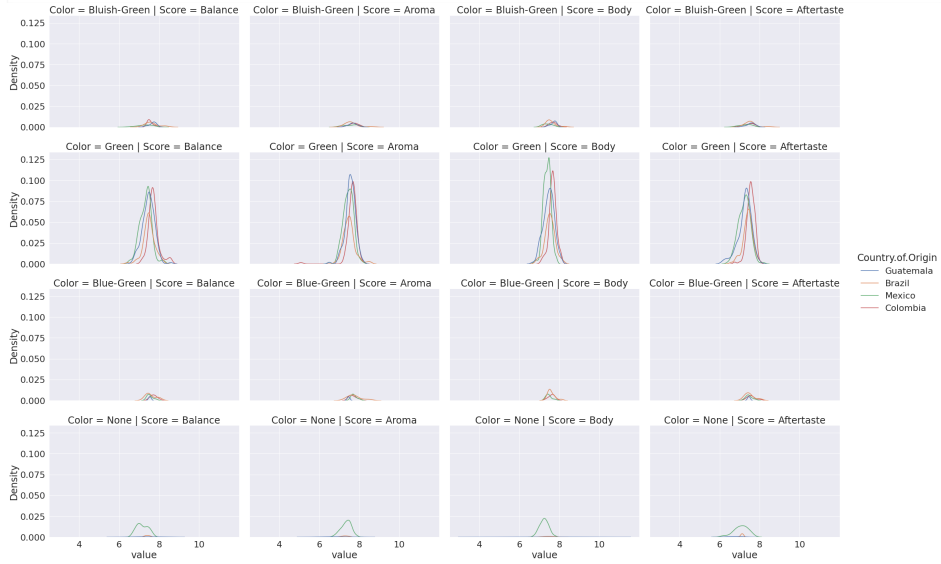

```
sns.displot(data= high_prod_coffee_df_melted,
            x='value',hue='Color',
            col = 'Score', col_wrap=2, kind='kde',aspect =1.5)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb764ff7e90>
```


```
sns.displot(data= high_prod_coffee_df_melted,
            x='value',hue='Country.of.Origin',
            col = 'Score', row='Color', kind='kde',aspect =1.5)
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/seaborn/distributions.py:316: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb764516a90>
```



10.1. Unpacking Jsons

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

```
course_gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
course_gh_df.head()
```

	id	type	actor	repo	payload	p
0	18280412476	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 400283911, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 8078961091, 'size': 2, 'distinct_s...	
1	18275823085	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 400656699, 'name': 'rhodyprog4ds/rhodyd...	{'push_id': 8076747374, 'size': 1, 'distinct_s...	
2	18274960415	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 400283911, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 8076346054, 'size': 1, 'distinct_s...	
3	18274903740	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 400283911, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 8076319428, 'size': 1, 'distinct_s...	
4	18274708793	IssuesEvent	{'id': 52358324, 'login': 'EvanRussellTheCoder...	{'id': 400283911, 'name': 'rhodyprog4ds/BrownF...	{'action': 'closed', 'issue': {'url': 'https:/...	

We want to transform each one of those from a dictionary like thing into a row in a data frame.

```
type(course_gh_df['actor'])
```

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

Recall, that base python types can be used as function, to cast an object from type to another.

```
5
```

```
5
```

```
type(5)
```

```
int
```

```
str(5)
```

```
'5'
```

To unpack one column we can cast each element of the column to a series and then stack them back together.

First, let's look at one row of one column

```
course_gh_df['actor'][0]
```

```
{'id': 10656079,
 'login': 'brownsarahm',
 'display_login': 'brownsarahm',
 'gravatar_id': '',
 'url': 'https://api.github.com/users/brownsarahm',
 'avatar_url': 'https://avatars.githubusercontent.com/u/10656079?'}
```

Now let's cast it to a Series

```
pd.Series(course_gh_df['actor'][0])
```

```
id                                                  10656079
login                                           brownsarahm
display_login                                   brownsarahm
gravatar_id
url                        https://api.github.com/users/brownsarahm
avatar_url       https://avatars.githubusercontent.com/u/10656079?
dtype: object
```

What we want is to do this over and over and stack them.

The `apply` method does this for us, in one compact step.

```
course_gh_df['actor'].apply(pd.Series)
```

	id	login	display_login	gravatar_id	
0	10656079	brownsarahm	brownsarahm		https://api.githu
1	10656079	brownsarahm	brownsarahm		https://api.githu
2	41898282	github-actions[bot]	github-actions		https://api.github.cc
3	10656079	brownsarahm	brownsarahm		https://api.githu
4	52358324	EvanRussellTheCoder	EvanRussellTheCoder		https://api.github.com/u
5	41898282	github-actions[bot]	github-actions		https://api.github.cc
6	10656079	brownsarahm	brownsarahm		https://api.githu
7	41898282	github-actions[bot]	github-actions		https://api.github.cc
8	10656079	brownsarahm	brownsarahm		https://api.githu
9	41898282	github-actions[bot]	github-actions		https://api.github.cc
10	10656079	brownsarahm	brownsarahm		https://api.githu
11	41898282	github-actions[bot]	github-actions		https://api.github.cc
12	10656079	brownsarahm	brownsarahm		https://api.githu
13	10656079	brownsarahm	brownsarahm		https://api.githu
14	41898282	github-actions[bot]	github-actions		https://api.github.cc
15	10656079	brownsarahm	brownsarahm		https://api.githu
16	41898282	github-actions[bot]	github-actions		https://api.github.cc
17	10656079	brownsarahm	brownsarahm		https://api.githu
18	41898282	github-actions[bot]	github-actions		https://api.github.cc
19	10656079	brownsarahm	brownsarahm		https://api.githu
20	41898282	github-actions[bot]	github-actions		https://api.github.cc
21	10656079	brownsarahm	brownsarahm		https://api.githu
22	41898282	github-actions[bot]	github-actions		https://api.github.cc
23	10656079	brownsarahm	brownsarahm		https://api.githu
24	41898282	github-actions[bot]	github-actions		https://api.github.cc
25	10656079	brownsarahm	brownsarahm		https://api.githu
26	10656079	brownsarahm	brownsarahm		https://api.githu
27	10656079	brownsarahm	brownsarahm		https://api.githu
28	41898282	github-actions[bot]	github-actions		https://api.github.cc
29	41898282	github-actions[bot]	github-actions		https://api.github.cc

How can we do this for all of the columns and put them back together after?

First, let's make a list of the columns we need to convert.

```
js_cols = ['actor','repo','payload','org']
```

When we use `.apply(pd.Series)` we get a a DataFrame.

```
type(course_gh_df['actor'].apply(pd.Series))
```

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

`pd.concat` takes a list of DataFrames and puts the together in one DataFrame.

to illustrate, it's nice to make small dataFrames.

```
df1 = pd.DataFrame([[1,2,3],[3,4,7]], columns = ['A','B','t'])
df2 = pd.DataFrame([[10,20,30],[30,40,70]], columns = ['AA','BB','t'])
df1
```

	A	B	t
0	1	2	3
1	3	4	7

```
df2
```

	AA	BB	t
0	10	20	30
1	30	40	70

If we use concat with the default settings, it stacks them vertically and aligns any columns that have the same name.

```
pd.concat([df1,df2])
```

	A	B	t	AA	BB
0	1.0	2.0	3	NaN	NaN
1	3.0	4.0	7	NaN	NaN
0	NaN	NaN	30	10.0	20.0
1	NaN	NaN	70	30.0	40.0

So, since the original DataFrames were both 2 rows with 3 columns each, with one column name appearing in both, we end up with a new DataFrame with shape (4,5) and it fills with `NaN` in the top right and the bottom left.

```
pd.concat([df1,df2]).shape
```

```
(4, 5)
```

We can use the `axis` parameter to tell it how to combine them. The default is `axis=0`, but `axis=1` will combine along rows.

```
pd.concat([df1,df2], axis =1)
```

	A	B	t	AA	BB	t
0	1	2	3	10	20	30
1	3	4	7	30	40	70

So now we get no NaN values, because both DataFrames have the same number of rows and the same index.

```
df1.index == df2.index
```

```
array([ True,  True])
```

and we have a total of 6 columns and 2 rows.

```
pd.concat([df1,df2], axis =1).shape
```

```
(2, 6)
```

Back to our gh data, we want to make a list of DataFrames where each DataFrame corresponds to one of the columns in the original DataFrame, but unpacked and then stack them horizontally (`axis=1`) because each DataFrame in the list is based on the same original DataFrame, they again have the same index.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=1)
```

	id	login	display_login	gravatar_id	
0	10656079	brownsarahm	brownsarahm		https://api.githu
1	10656079	brownsarahm	brownsarahm		https://api.githu
2	41898282	github-actions[bot]	github-actions		https://api.github.co
3	10656079	brownsarahm	brownsarahm		https://api.githu
4	52358324	EvanRussellTheCoder	EvanRussellTheCoder		https://api.github.com/u
5	41898282	github-actions[bot]	github-actions		https://api.github.co
6	10656079	brownsarahm	brownsarahm		https://api.githu
7	41898282	github-actions[bot]	github-actions		https://api.github.co
8	10656079	brownsarahm	brownsarahm		https://api.githu
9	41898282	github-actions[bot]	github-actions		https://api.github.co
10	10656079	brownsarahm	brownsarahm		https://api.githu
11	41898282	github-actions[bot]	github-actions		https://api.github.co
12	10656079	brownsarahm	brownsarahm		https://api.githu
13	10656079	brownsarahm	brownsarahm		https://api.githu
14	41898282	github-actions[bot]	github-actions		https://api.github.co
15	10656079	brownsarahm	brownsarahm		https://api.githu
16	41898282	github-actions[bot]	github-actions		https://api.github.co
17	10656079	brownsarahm	brownsarahm		https://api.githu
18	41898282	github-actions[bot]	github-actions		https://api.github.co
19	10656079	brownsarahm	brownsarahm		https://api.githu
20	41898282	github-actions[bot]	github-actions		https://api.github.co
21	10656079	brownsarahm	brownsarahm		https://api.githu
22	41898282	github-actions[bot]	github-actions		https://api.github.co
23	10656079	brownsarahm	brownsarahm		https://api.githu
24	41898282	github-actions[bot]	github-actions		https://api.github.co
25	10656079	brownsarahm	brownsarahm		https://api.githu
26	10656079	brownsarahm	brownsarahm		https://api.githu
27	10656079	brownsarahm	brownsarahm		https://api.githu
28	41898282	github-actions[bot]	github-actions		https://api.github.co
29	41898282	github-actions[bot]	github-actions		https://api.github.co

30 rows × 23 columns

Try it Yourself

examine the list of DataFrames to see what structure they share and do not share

In this case we get the same 30 rows, beacuse that's what the API gave us and turned our 4 columns from `js_cols` into 26 columns.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=1).shape
```

```
(30, 23)
```

If we had used the default, we'd end up with 120 rows (30*4) and we have only 19 columns, because there are subfield names that are shared across the original columns. (eg most have an `id`)

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=0).shape
```

```
(120, 16)
```

Try it yourself

How could you anticipate how many are shared?

we might want to rename the new columns so that they have the original column name prepended to the new name. This will help us distinguish between the different `id` columns

pandas has a `rename` method for this.

and this is another job for lambdas.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series).rename(columns = lambda c: cur_col +
'_' +c)
             for cur_col in js_cols],
          axis=1)
```

	actor_id	actor_login	actor_display_login	actor_gravatar_id	
0	10656079	brownsarahm	brownsarahm		https://a
1	10656079	brownsarahm	brownsarahm		https://a
2	41898282	github-actions[bot]	github-actions		https://api.gi
3	10656079	brownsarahm	brownsarahm		https://a
4	52358324	EvanRussellTheCoder	EvanRussellTheCoder		https://api.github
5	41898282	github-actions[bot]	github-actions		https://api.gi
6	10656079	brownsarahm	brownsarahm		https://a
7	41898282	github-actions[bot]	github-actions		https://api.gi
8	10656079	brownsarahm	brownsarahm		https://a
9	41898282	github-actions[bot]	github-actions		https://api.gi
10	10656079	brownsarahm	brownsarahm		https://a
11	41898282	github-actions[bot]	github-actions		https://api.gi
12	10656079	brownsarahm	brownsarahm		https://a
13	10656079	brownsarahm	brownsarahm		https://a
14	41898282	github-actions[bot]	github-actions		https://api.gi
15	10656079	brownsarahm	brownsarahm		https://a
16	41898282	github-actions[bot]	github-actions		https://api.gi
17	10656079	brownsarahm	brownsarahm		https://a
18	41898282	github-actions[bot]	github-actions		https://api.gi
19	10656079	brownsarahm	brownsarahm		https://a
20	41898282	github-actions[bot]	github-actions		https://api.gi
21	10656079	brownsarahm	brownsarahm		https://a
22	41898282	github-actions[bot]	github-actions		https://api.gi
23	10656079	brownsarahm	brownsarahm		https://a
24	41898282	github-actions[bot]	github-actions		https://api.gi
25	10656079	brownsarahm	brownsarahm		https://a
26	10656079	brownsarahm	brownsarahm		https://a
27	10656079	brownsarahm	brownsarahm		https://a
28	41898282	github-actions[bot]	github-actions		https://api.gi
29	41898282	github-actions[bot]	github-actions		https://api.gi

30 rows × 23 columns

the `rename` method's `column` parameter can take a lambda defined inline, which is helpful, because we want that function to take one parameter (the current columnt name) and do the same thing to all of the columns within a single DataFrame, but to prepend a different thing for each DataFrame

```
pd.concat([course_gh_df[cur_col].apply(pd.Series).rename(columns = lambda c: cur_col +
'_' +c)
           for cur_col in js_cols],
          axis=1)
```

	actor_id	actor_login	actor_display_login	actor_gravatar_id	
0	10656079	brownsarahm	brownsarahm		https://a
1	10656079	brownsarahm	brownsarahm		https://a
2	41898282	github-actions[bot]	github-actions		https://api.gi
3	10656079	brownsarahm	brownsarahm		https://a
4	52358324	EvanRussellTheCoder	EvanRussellTheCoder		https://api.github
5	41898282	github-actions[bot]	github-actions		https://api.gi
6	10656079	brownsarahm	brownsarahm		https://a
7	41898282	github-actions[bot]	github-actions		https://api.gi
8	10656079	brownsarahm	brownsarahm		https://a
9	41898282	github-actions[bot]	github-actions		https://api.gi
10	10656079	brownsarahm	brownsarahm		https://a
11	41898282	github-actions[bot]	github-actions		https://api.gi
12	10656079	brownsarahm	brownsarahm		https://a
13	10656079	brownsarahm	brownsarahm		https://a
14	41898282	github-actions[bot]	github-actions		https://api.gi
15	10656079	brownsarahm	brownsarahm		https://a
16	41898282	github-actions[bot]	github-actions		https://api.gi
17	10656079	brownsarahm	brownsarahm		https://a
18	41898282	github-actions[bot]	github-actions		https://api.gi
19	10656079	brownsarahm	brownsarahm		https://a
20	41898282	github-actions[bot]	github-actions		https://api.gi
21	10656079	brownsarahm	brownsarahm		https://a
22	41898282	github-actions[bot]	github-actions		https://api.gi
23	10656079	brownsarahm	brownsarahm		https://a
24	41898282	github-actions[bot]	github-actions		https://api.gi
25	10656079	brownsarahm	brownsarahm		https://a
26	10656079	brownsarahm	brownsarahm		https://a
27	10656079	brownsarahm	brownsarahm		https://a
28	41898282	github-actions[bot]	github-actions		https://api.gi
29	41898282	github-actions[bot]	github-actions		https://api.gi

30 rows × 23 columns

So now, we have the unpacked columns with good column names, but we lost the columns that were originally good.

How can we append the new columns to the old ones? First we can make a DataFrame that's just the columns not on the list that we're goign to expand.

```
course_gf_df_good = course_gh_df[[col for col in
          course_gh_df.columns if not(col in js_cols)]]
course_gf_df_good
```

	id	type	public	created_at
0	18280412476	PushEvent	True	2021-10-05 02:03:20+00:00
1	18275823085	PushEvent	True	2021-10-04 18:41:19+00:00
2	18274960415	PushEvent	True	2021-10-04 17:40:46+00:00
3	18274903740	PushEvent	True	2021-10-04 17:36:45+00:00
4	18274708793	IssuesEvent	True	2021-10-04 17:22:57+00:00
5	18249707673	PushEvent	True	2021-10-02 02:07:50+00:00
6	18249692276	PushEvent	True	2021-10-02 02:04:08+00:00
7	18249671571	PushEvent	True	2021-10-02 01:59:13+00:00
8	18249657039	PushEvent	True	2021-10-02 01:55:24+00:00
9	18248222325	PushEvent	True	2021-10-01 21:52:39+00:00
10	18248190104	PushEvent	True	2021-10-01 21:48:46+00:00
11	18248128603	PushEvent	True	2021-10-01 21:41:33+00:00
12	18248091131	PushEvent	True	2021-10-01 21:37:14+00:00
13	18223008981	PushEvent	True	2021-09-30 12:53:49+00:00
14	18214901041	PushEvent	True	2021-09-30 02:09:52+00:00
15	18214863649	PushEvent	True	2021-09-30 02:05:30+00:00
16	18214403927	PushEvent	True	2021-09-30 01:07:17+00:00
17	18214360702	PushEvent	True	2021-09-30 01:02:18+00:00
18	18213077567	PushEvent	True	2021-09-29 22:27:40+00:00
19	18213046815	PushEvent	True	2021-09-29 22:24:17+00:00
20	18212484236	PushEvent	True	2021-09-29 21:30:45+00:00
21	18212438695	PushEvent	True	2021-09-29 21:26:37+00:00
22	18187502311	PushEvent	True	2021-09-28 14:42:58+00:00
23	18187438407	PushEvent	True	2021-09-28 14:39:37+00:00
24	18174833999	PushEvent	True	2021-09-28 00:36:32+00:00
25	18174790150	PushEvent	True	2021-09-28 00:33:07+00:00
26	18174067757	PushEvent	True	2021-09-27 23:37:00+00:00
27	18173918708	PushEvent	True	2021-09-27 23:25:35+00:00
28	18172434203	PushEvent	True	2021-09-27 21:40:50+00:00
29	18172250315	PushEvent	True	2021-09-27 21:28:46+00:00

Then we can prepend that to the list that we pass to `concat`. We have to put it in a list first, then use + to do that.

```
pd.concat([course_gf_df_good]+[course_gh_df[col].apply(pd.Series,).rename(
  columns= lambda i_col: col + '_' + i_col )
      for col in js_cols],axis=1)
```

	id	type	public	created_at	actor_id	actor_login	act
0	18280412476	PushEvent	True	2021-10-05 02:03:20+00:00	10656079	brownsarahm	
1	18275823085	PushEvent	True	2021-10-04 18:41:19+00:00	10656079	brownsarahm	
2	18274960415	PushEvent	True	2021-10-04 17:40:46+00:00	41898282	github-actions[bot]	
3	18274903740	PushEvent	True	2021-10-04 17:36:45+00:00	10656079	brownsarahm	
4	18274708793	IssuesEvent	True	2021-10-04 17:22:57+00:00	52358324	EvanRussellTheCoder	Evan
5	18249707673	PushEvent	True	2021-10-02 02:07:50+00:00	41898282	github-actions[bot]	
6	18249692276	PushEvent	True	2021-10-02 02:04:08+00:00	10656079	brownsarahm	
7	18249671571	PushEvent	True	2021-10-02 01:59:13+00:00	41898282	github-actions[bot]	
8	18249657039	PushEvent	True	2021-10-02 01:55:24+00:00	10656079	brownsarahm	
9	18248222325	PushEvent	True	2021-10-01 21:52:39+00:00	41898282	github-actions[bot]	
10	18248190104	PushEvent	True	2021-10-01 21:48:46+00:00	10656079	brownsarahm	
11	18248128603	PushEvent	True	2021-10-01 21:41:33+00:00	41898282	github-actions[bot]	
12	18248091131	PushEvent	True	2021-10-01 21:37:14+00:00	10656079	brownsarahm	
13	18223008981	PushEvent	True	2021-09-30 12:53:49+00:00	10656079	brownsarahm	
14	18214901041	PushEvent	True	2021-09-30 02:09:52+00:00	41898282	github-actions[bot]	
15	18214863649	PushEvent	True	2021-09-30 02:05:30+00:00	10656079	brownsarahm	
16	18214403927	PushEvent	True	2021-09-30 01:07:17+00:00	41898282	github-actions[bot]	
17	18214360702	PushEvent	True	2021-09-30 01:02:18+00:00	10656079	brownsarahm	
18	18213077567	PushEvent	True	2021-09-29 22:27:40+00:00	41898282	github-actions[bot]	
19	18213046815	PushEvent	True	2021-09-29 22:24:17+00:00	10656079	brownsarahm	
20	18212484236	PushEvent	True	2021-09-29 21:30:45+00:00	41898282	github-actions[bot]	
21	18212438695	PushEvent	True	2021-09-29 21:26:37+00:00	10656079	brownsarahm	
22	18187502311	PushEvent	True	2021-09-28 14:42:58+00:00	41898282	github-actions[bot]	
23	18187438407	PushEvent	True	2021-09-28 14:39:37+00:00	10656079	brownsarahm	
24	18174833999	PushEvent	True	2021-09-28 00:36:32+00:00	41898282	github-actions[bot]	
25	18174790150	PushEvent	True	2021-09-28 00:33:07+00:00	10656079	brownsarahm	
26	18174067757	PushEvent	True	2021-09-27 23:37:00+00:00	10656079	brownsarahm	
27	18173918708	PushEvent	True	2021-09-27 23:25:35+00:00	10656079	brownsarahm	
28	18172434203	PushEvent	True	2021-09-27 21:40:50+00:00	41898282	github-actions[bot]	
29	18172250315	PushEvent	True	2021-09-27 21:28:46+00:00	41898282	github-actions[bot]	

30 rows × 27 columns

To see how the list math works

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

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

results in one list

but without the `[]` we get a type error

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_1846/858457300.py in <module>
----> 1 'a' + ['b','c','d']

TypeError: can only concatenate str (not "list") to str
```

List operations return `None` and mutate the list in place so

```
orig_list = ['a']
new_items = ['b','c','d']
orig_list.extend(new_items)
```

outputs nothing because `None` was returned and it changes the original variable.

```
orig_list
```

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

```
type(orig_list.extend(new_items))
```

```
NoneType
```

is none.

10.2. Questions After Class

All clarifying questions today

10.2.1. How does Axis work?

10.2.2. How does melt work?

10.2.3. What about the NaNs that are still left?

11. Missing Data and Inconsistent coding

```
import pandas as pd
import seaborn as sns
import numpy as np #
na_toy_df = pd.DataFrame(data = [[1,3,4,5],[2 ,6, np.nan]])

# make plots look nicer and increase font size
sns.set_theme(font_scale=2)
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)


rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
course_gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
```

So far, we've dealt with structural issues in data. but there's a lot more to cleaning.

Today, we'll deal with how to fix the values wihtin the data. To see the types of things:

Stanford Policy Lab Open Policing Project data readme Propublica Machine Bias the "How we acquired data" section

11.1. Missing Values

Dealing with missing data is a whole research area. There isn't one solution.

in 2020 there was a workshop on it

There are also many classic approaches both when training and when applying models.

example application in breast cancer detection

In pandas, even representing missing values is under experimentation. Currently, it uses `numpy.NaN`, but the experiment is with `pd.NA`.

Missing values even causes the datatypes to change

Pandas gives a few basic tools:

- drop with (`dropna`)
- fill with `fillna`

```
coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc

5 rows × 44 columns

The 'Lot.Number' has a lot of NaN values, how can we explore it?

We can look at the type:

```
coffee_df['Lot.Number'].dtype
```

```
dtype('O')
```

And we can look at the value counts.

```
coffee_df['Lot.Number'].value_counts()
```

```
1                             18
020/17                         6
019/17                         5
2                              3
102                            3
                              ..
11/23/0696                     1
3-59-2318                      1
8885                           1
5055                           1
017-053-0211/ 017-053-0212     1
Name: Lot.Number, Length: 221, dtype: int64
```

Filling can be good if you know how to fill reasonably, but don't have data to spare by dropping. For example

- you can approximate with another column
- you can approximate with that column from other rows

We see that a lot are '1', maybe we know that when the data was collected, if the Farm only has one lot, some people recorded '1' and others left it as missing. So we could fill in with 1:

```
coffee_df['Lot.Number'].fillna(1).head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: Lot.Number, dtype: object
```

```
coffee_df['Lot.Number'].head()
```

```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
Name: Lot.Number, dtype: object
```

Tip

Note that even after we called `fillna` we display it again and the original data is unchanged.

To save the filled in column we have a few choices:

- use the `inplace` parameter. This doesn't offer performance advantages, but does It still copies the object, but then reassigns the pointer. Its under discussion to deprecate
- write to a new DataFrame
- add a column

We'll use adding a column:

```
coffee_df['lot_number_clean'] = coffee_df['Lot.Number'].fillna(1)
```

Question in Class

When I use value counts it treats the filled ones as different. Why?

```
coffee_df['Lot.Number'].value_counts()
```

```
1                             18
020/17                         6
019/17                         5
2                              3
102                            3
                              ..
11/23/0696                     1
3-59-2318                      1
8885                           1
5055                           1
017-053-0211/ 017-053-0212     1
Name: Lot.Number, Length: 221, dtype: int64
```

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

```
1                             1041
1                               18
020/17                           6
019/17                           5
102                              3
                               ...
3-59-2318                        1
8885                             1
5055                             1
MCCFWXA15/16                     1
017-053-0211/ 017-053-0212       1
Name: lot_number_clean, Length: 222, dtype: int64
```

If we swithc to `1` as a string, then we'd see all of the one values as the same thing.

```
coffee_df['lot_number_clean'] = coffee_df['Lot.Number'].fillna('1')

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

```
1                             1059
020/17                           6
019/17                           5
102                              3
103                              3
                               ...
3-59-2318                        1
8885                             1
5055                             1
MCCFWXA15/16                     1
017-053-0211/ 017-053-0212       1
Name: lot_number_clean, Length: 221, dtype: int64
```

This was our goal, so in this case, it's the right thing to do to overwrite the value.

Dropping is a good choice when you otherwise have a lot of data and the data is missing at random.

Dropping can be risky if it's not missing at random. For example, if we saw in the coffee data that one of the scores was missing for all of the rows from one country, or even just missing more often in one country, that could bias our results.

To illustrate how `dropna` works, we'll use the `shape` method:

```
coffee_df.shape
```

```
(1311, 45)
```

By default, it drops any row with one or more `NaN` values.

```
coffee_df.dropna().shape
```

```
(130, 45)
```

We could instead tell it to only drop rows with `NaN` in a subset of the columns.

```
coffee_df.dropna(subset=['altitude_low_meters']).shape
```

```
(1084, 45)
```

whatever you do, document it

In the Open Policing Project Data Summary we saw that they made a summary information that showed which variables had at least 70% not missing values. We can similarly choose to keep only variables that have more than a specific threshold of data, using the `thresh` parameter and `axis=1` to drop along columns.

```
n_rows, n_cols = coffee_df.shape

coffee_df.dropna(thresh=.7*n_rows,axis=1).shape
```

```
(1311, 44)
```

This dataset is actually in pretty good shape, but if we use a more stringent threshold it drops more columns.

```
coffee_df.dropna(thresh=.85*n_rows,axis=1).shape
```

```
(1311, 34)
```

Important

Everthhting after this is new material that we did not have time for in class, but is important and helpful in your assignment (and for your portflio).

11.2. Inconsistent values

This was one of the things that many of you anticipated or had observed. A useful way to investigate for this, is to use `value_counts` and sort them alphabetically by the values from the original data, so that similar ones will be consecutive in the list. Once we have the `value_counts()` Series, the values from the `coffee_df` become the index, so we use `sort_index`.

Let's look at the `In.Country.Partner` column

```
coffee_df['In.Country.Partner'].value_counts().sort_index()
```

Note

subset operates along columns by default, because axis is set to 0, by default.

```
AMECAFE
205
Africa Fine Coffee Association
49
Almacafé
178
Asociacion Nacional Del Café
155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.
6
Asociación de Cafés Especiales de Nicaragua
8
Blossom Valley International
58
Blossom Valley International\n
1
Brazil Specialty Coffee Association
67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao
1
Centro Agroecológico del Café A.C.
8
Coffee Quality Institute
7
Ethiopia Commodity Exchange
18
Instituto Hondureño del Café
60
Kenya Coffee Traders Association
22
METAD Agricultural Development plc
15
NUCOFFEE
36
Salvadoran Coffee Council
11
Specialty Coffee Ass
1
Specialty Coffee Association
295
Specialty Coffee Association of Costa Rica
42
Specialty Coffee Association of Indonesia
10
Specialty Coffee Institute of Asia
16
Tanzanian Coffee Board
6
Torch Coffee Lab Yunnan
2
Uganda Coffee Development Authority
22
Yunnan Coffee Exchange
12
Name: In.Country.Partner, dtype: int64
```

We can see there's only one `Blossom Valley International\n` but 58 `Blossom Valley International`, the former is likely a typo, especially since `\n` is a special character for a newline. Similarly, with 'Specialty Coffee Ass' and 'Specialty Coffee Association'.

This is another job for dictionaries, we make one with the value to replace as the key and the value to insert as the value.

```
partner_corrections = {'Blossom Valley International\n':'Blossom Valley International',
  'Specialty Coffee Ass':'Specialty Coffee Association'}
coffee_df['in_country_partner_clean'] = coffee_df['In.Country.Partner'].replace(
  to_replace=partner_corrections)
coffee_df['in_country_partner_clean'].value_counts().sort_index()
```

```
AMECAFE
205
Africa Fine Coffee Association
49
Almacafé
178
Asociacion Nacional Del Café
155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.
6
Asociación de Cafés Especiales de Nicaragua
8
Blossom Valley International
59
Brazil Specialty Coffee Association
67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao
1
Centro Agroecológico del Café A.C.
8
Coffee Quality Institute
7
Ethiopia Commodity Exchange
18
Instituto Hondureño del Café
60
Kenya Coffee Traders Association
22
METAD Agricultural Development plc
15
NUCOFFEE
36
Salvadoran Coffee Council
11
Specialty Coffee Association
296
Specialty Coffee Association of Costa Rica
42
Specialty Coffee Association of Indonesia
10
Specialty Coffee Institute of Asia
16
Tanzanian Coffee Board
6
Torch Coffee Lab Yunnan
2
Uganda Coffee Development Authority
22
Yunnan Coffee Exchange
12
Name: in_country_partner_clean, dtype: int64
```

and now we see the corrected values. We can also pass lambdas or put lambas in the dictionary if there are systemic patterns.

11.3. Fixing data at load time

Explore some of the different parameters in `read_csv`

How can we read in data that looks like this:

Ethnicity	Asian				Black				Hispanic			
Gender	Female		Male		Female		Male		Female		Male	
	count	mean	count	mean	count	mean	count	mean	count	mean	count	m
Age Cohort												
0 - 5	6	1544.33333							18	1431.33333	26	1366
51 +												

```
pd.read_csv('fancy_formatting.xlsx', header = list(range(4)))
```

Many problems can be repaired with parameters in read_csv.

11.4. A Cleaning Data Recipe

not everything possible, but good enough for this course

1. Can you use parameters to read the data in better?

2. Fix the index and column headers (making these easier to use makes the rest easier)
3. Is the data structured well?
4. Are there missing values?
5. Do the datatypes match what you expect by looking at the head or a sample?
6. Are categorical variables represented in usable way?
7. Does your analysis require filtering or augmenting the data?

Things to keep in mind:

- always save new copies of data when you mutate it
- add new columns rather than overwriting columns
- long variable names are better than ambiguous naming

11.5. Your observations from Monday:

I promised we'd come back to your observations on what problems could occur in data. Here they are, organized by rough categories of when/how to fix them.

We can fix while reading in data:

- decimal was indicated with ',' insead of '.' so pandas saw value as a string rather than a float
- missing header
- reading the index as a column
- large datasets might be too slow or not fit in memory
- missing data represeted with a value or special character

We can fix by reshaping data:

- Data can get read into tables in bizarre ways depending on how the data was entered originally.
- every value in one column, instead of separated

We can repair by changing values or filtering:

- information represented inconsistently eg "Value" and " Value " or twenty-two instead of 22
- blank rows or blank columns or data that is N/A
- date/time information can be represented lots of different ways
- representing categorical with numbers that are ambiguous
- spaces or other symbols in column names
- some numbers as strings, others as ints within a column
- symbols being mis interpreted

Real problems, but beyond our scope:

- corrupt data files

11.6. More Practice

Instead of more practice with these manipulations, below are more examples of cleaning data to see how these types of manipulations get used.
Your goal here is not to memorize every possible thing, but to build a general idea of what good data looks like and good habits for cleaning data and keeping it reproducible.

- Cleaning the Adult Dataset
- All Shades

Also here are some tips on general data management and organization.

This article is a comprehensive discussion of data cleaning.

12. Building Datasets From multiple Sources

1. jsdkfds
2. fdsf
3. kdskfjsld

```
import pandas as pd
```

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

Today we're going to look at some data on NBA (National Basketball Association) players.

12.1. Combining Multiple Tables

```
p18 = pd.read_csv(course_data_url +'2018-players.csv')
p19 = pd.read_csv(course_data_url +'2019-players.csv')
```

```
p18.head()
```

	TEAM_ID	PLAYER_ID	SEASON
0	1610612761	202695	2018
1	1610612761	1627783	2018
2	1610612761	201188	2018
3	1610612761	201980	2018
4	1610612761	200768	2018

12.1.1. Stacking with Concat

We've seen one way of putting dataframes together with `concat` we used it when Unpacking Jsons. In that case, we stacked them side by side, now we want to stack them vertically, which is the default of concat.

```
players_df = pd.concat([p18,p19])
players_df.head()
```

	TEAM_ID	PLAYER_ID	SEASON	PLAYER_NAME
0	1610612761	202695	2018	NaN
1	1610612761	1627783	2018	NaN
2	1610612761	201188	2018	NaN
3	1610612761	201980	2018	NaN
4	1610612761	200768	2018	NaN

This has the same columns, and we can see what happened more by looking at the shape of each.

```
players_df.shape
```

```
(1374, 4)
```

```
p18.shape, p19.shape
```

```
((748, 3), (626, 4))
```

We can verify that the length of the new data frame is the sum of the original two DataFrames.

```
assert len(p18) + len(p19) == len(players_df)
```

12.1.2. Combining Data with Merge

What is we want to see which players changed teams from 2018 to 2019? We can do this with `merge`. For `merge` we have to tell it a left DataFrame and a right DataFrame and on what column to match them up.

The left and right DataFrames will be used different ways, but any DataFrame can be put in either position.

For this case, we will use 2018 data as left ,and 2019 as right and then merge `on='PLAYER_ID'`.

```
pd.merge(p18,p19,on='PLAYER_ID').head()
```

	TEAM_ID_x	PLAYER_ID	SEASON_x	PLAYER_NAME	TEAM_ID_y	SEASON_y
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019
2	1610612761	201188	2018	Marc Gasol	1610612761	2019
3	1610612763	201188	2018	Marc Gasol	1610612761	2019
4	1610612761	201980	2018	Danny Green	1610612747	2019

Now, we get what we expect, but the column names have `_x` and `_y` on the end (as a suffix, appended to the original). We'll add 2018 and 2019 respectively, separated with a `_`.

```
year_over_year = pd.merge(p18,p19,on='PLAYER_ID',suffixes=('_2018','_2019'))
year_over_year.head()
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2
2	1610612761	201188	2018	Marc Gasol	1610612761	2
3	1610612763	201188	2018	Marc Gasol	1610612761	2
4	1610612761	201980	2018	Danny Green	1610612747	2

Now that it's a little bit cleaner, we will examine how it works by looking at the shape.

```
year_over_year.shape, p18.shape, p19.shape
```

```
((538, 6), (748, 3), (626, 4))
```

This kept only the players that played both years, with repetitions for each team they played on for each year.

We can check the calculation with set math (python `set` type has operations from math sets like intersect and set difference)

```
# player IDs for each year, no repeats
p19_u = set(p19['PLAYER_ID'])
p18_u = set(p18['PLAYER_ID'])

# player IDs that played both years
p1819 = p18_u.intersection(p19_u)

# teams per player per year
teams_per_p18 = p18['PLAYER_ID'].value_counts()
teams_per_p19 = p19['PLAYER_ID'].value_counts()

# total number of team-player combinations
# multiply number of teams each player played for in 18 by number of teams in 19
# then sum.  (most of of these are 1*1)
sum(teams_per_p19[p1819]* teams_per_p18[p1819])
```

```
538
```

We can also merge so that we keep all players on either team using `how='outer'` the default value for `how` is inner, which takes the intersection (but with duplicates, does some extra things as we saw). With `outer` it takes the union, but with extra handling for the duplicates.

```
pd.merge(p18,p19,on='PLAYER_ID',suffixes=('_2018','_2019'),how='outer').shape
```

```
(927, 6)
```

It's the total of the rows we had before, plus the total number of player-teams for players that only played in one of the two years.

```
#players tha tonly played in one year
o18 = p18_u.difference(p19_u)
o19 = p19_u.difference(p18_u)
# teams those players played for + the above 538
teams_per_p19[o19].sum() + teams_per_p18[o18].sum() + sum(teams_per_p19[p1819]*
teams_per_p18[p1819])
```

```
927
```

We can save this to a variable

```
year_over_year_outer =  pd.merge(p18,p19,on='PLAYER_ID',suffixes=
('_2018','_2019'),how='outer')
```

then look at a few rows to see that it has indeed filled in with NaN in the places where there wasn't a value.

```
year_over_year_outer.sample(10)
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON
681	1.610613e+09	202918	2018.0	NaN	NaN	
241	1.610613e+09	1626167	2018.0	Myles Turner	1.610613e+09	2
271	1.610613e+09	1629353	2018.0	Isaac Humphries	1.610613e+09	2
208	1.610613e+09	1629033	2018.0	Theo Pinson	1.610613e+09	2
742	1.610613e+09	1627875	2018.0	NaN	NaN	
395	1.610613e+09	1628249	2018.0	NaN	NaN	
582	1.610613e+09	201950	2018.0	Jrue Holiday	1.610613e+09	2
226	1.610613e+09	201961	2018.0	Wayne Ellington	1.610613e+09	2
443	1.610613e+09	203546	2018.0	NaN	NaN	
495	1.610613e+09	202688	2018.0	Brandon Knight	1.610613e+09	2

Note

We can also tell that there are NaN beacuse it cast the year to float from int.

12.2. Merge types, in detail

We can examine how these things work more visually with smaller DataFrames:

```
left = pd.DataFrame(
  {
    "key1": ["K0", "K0", "K1", "K2"],
    "key2": ["K0", "K1", "K0", "K1"],
    "A": ["A0", "A1", "A2", "A3"],
    "B": ["B0", "B1", "B2", "B3"],
  }
)


right = pd.DataFrame(
  {
    "key1": ["K0", "K1", "K1", "K2"],
    "key2": ["K0", "K0", "K0", "K0"],
    "C": ["C0", "C1", "C2", "C3"],
    "D": ["D0", "D1", "D2", "D3"],
  }
)
```

```
left
```

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

```
right
```

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

```
pd.merge(left, right, on=["key1", "key2"], how='inner')
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Note

inner is default, but we can state it to be more explicit

```
pd.merge(left, right, on=["key1", "key2"], how='outer' )
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

12.3. Duplicate Keys

```
left = pd.DataFrame({"A": [1, 2], "B": [2, 2]})

right = pd.DataFrame({"A": [4, 5, 6], "B": [2, 2, 2]})

result = pd.merge(left, right, on="B", how="outer")
```

```
left
```

	A	B
0	1	2
1	2	2

```
right
```

	A	B
0	4	2
1	5	2
2	6	2

```
result
```

	A_x	B	A_y
0	1	2	4
1	1	2	5
2	1	2	6
3	2	2	4
4	2	2	5
5	2	2	6

If we ask pandas to validate the merge with `validate` and a specific type of merge, it will throw an error if that type of merge is not possible.

```
pd.merge(left, right,on='B',validate='one_to_one')
```

```
---------------------------------------------------------------------------
MergeError                                Traceback (most recent call last)
/tmp/ipykernel_1904/2916808787.py in <module>
----> 1 pd.merge(left, right,on='B',validate='one_to_one')

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/reshape/merge.py in merge(left, right, how, on, left_on, right_on,
left_index, right_index, sort, suffixes, copy, indicator, validate)
    116         copy=copy,
    117         indicator=indicator,
--> 118         validate=validate,
    119     )
    120     return op.get_result()

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/reshape/merge.py in __init__(self, left, right, how, on, left_on,
right_on, axis, left_index, right_index, sort, suffixes, copy, indicator, validate)
    706         # are in fact unique.
    707         if validate is not None:
--> 708             self._validate(validate)
    709
    710     def get_result(self) -> DataFrame:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/reshape/merge.py in _validate(self, validate)
   1418             if not left_unique and not right_unique:
   1419                 raise MergeError(
-> 1420                     "Merge keys are not unique in either left "
   1421                     "or right dataset; not a one-to-one merge"
   1422                 )

MergeError: Merge keys are not unique in either left or right dataset; not a one-to-one
merge
```

Further Reading

The pandas documentation is a good place to read through their exampels on how validate works. it's important to note the types of possible joins from the beginning of the section

12.4. Questions at the End of Class

12.4.1. how to remove certain columns in merges (i.e. the year columns in the basketball dataset since they are redundant)

12.4.2. Can we merge as many dataframes as we would want?

12.4.3. Is there no way to merge the left and right for the A and B data?

12.4.4. Can you use merging to check correlation between two different sets?

12.4.5. Is there any way to compare the team ids in both datasets so that it outputs the players that changed teams between those times?

12.4.6. in year_over_year, where suffixes=('_2018','_2019'), why is there an underscore?

12.4.7. If we merge two dataframes which each have a column (or columns) that are the same, but each have different data types in those columns, what happens?

12.4.8. how does suffix to know to change the date?

12.5. More Practice

1. Use a merge to figure out how many players did not return for the 2019 season
2. Also load the `conferences.csv` data. Use this to figure out how many players moved conferences from 2018 to 2019.

1. Portfolio Setup, Data Science, and Python

Due: 2020-09-12

1.1. Objective & Evaluation

This assignment is an opportunity to earn level 2 achievements for the `process` and `python` and confirm that you have all of your tools setup, including your portfolio.

1.2. To Do

> **Important**
>
> If you have trouble, check the GitHub FAQ on the left before e-mailing

````
```{warning}
If you have trouble with the (*)d steps, don't worry, we can help work around these later. To help
us out, document the errors as bugs on your repository.
```
````

> **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/fall21instructors
>
> To do this click Issues at the top, the green "New Issue" button and then type away.

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 change the logo if you wish
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. (*) Install some additional python packages with: `pip install pip install -r requirements.txt` (this is a python operation, so use anaconda prompt on Windows, if the pip version doesn't work, try it with conda: `conda install --file requirements.txt`) form inside the portfolio folder
7. (*) Configure precommit to help keep your repo clean with `pre-commit install`. If this step doesn't work, see the portfolio README under "Using your Jupyter Book Portfolio"
8. 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**
    - 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. Remember you will have many chances to earn level 2 achievement in python
9. 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 (+/-)
    '''
```

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

1.3. Submission Instructions

Create a Jupyter Notebook with your function in your portfolio folder 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.

There will be a pull request on your repository that is made by GitHub classroom, request a review from @rhodypro4dg/fall21instructors.

2. Practicing Python and Accessing Data

due : 2020-09-21

2.1. 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. The following table summarizes the grading. It supplements the skill definitions from the Achievement Definitions.

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 rubric for grading

First, accept the assignment . It contains a notebook with some template structure (and will set you up for grading).

2.2. 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 notebook that includes:

- heading of the dataset's name
- a link to where someone can learn about the dataset
- a 1-2 sentence summary of what the dataset contains and why it was collected
- 1-2 questions you would like to answer with that dataset.

Hint

The Datasets page has information about data for any assignment. For this assignment, the Best for loading directly into a notebook section is probably the best place to start.

2.3. Store them for loading

Create a list of dictionaries in `datasets.py`, so that there is one dictionary for each dataset with the url, a name, and what function should be used to load the data into a `pandas.DataFrame`.

Hint

The goal here is to set up this list of dictionaries so that you can load data using different functions in each pass through the loop, without an `if` statement. You'll be iterating over the list of dictionaries, so the loop variable be a different dictionary each time.

see the where we used a lambda in a dictionary in the class notes for more information.

Hint

Any `.py` file can become a module

Tip

Urls are strings. The `string` class in python has a lot of helpful methods for manipulating strings, like `split`.

2.4. 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. save it 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.
2. record attributes about the dataset as in the table below in a list of lists:
3. Use that to create a DataFrame with the following columns:

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 2.2 Meta Data Description of the DataFrame to build

> **Tip**
>
> The pandas IO page has information about how to read data in and save data out of pandas.

2.5. Manipulate your datasets

For one dataset that includes nonnumerical data:

- display the heading and the last 4 rows
- make and display a new data frame with only the numerical columns (select these programmatically)

For any other dataset:

- display the heading and 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:

- display the first 3 odd rows (eg 1,3,5) of the data for two columns of your choice

2.6. Exploring data files

For each dataset, in a separate section of your notebook titled `When things go wrong`:

- try reading in data with the wrong `read_` function and make notes about what happens.
- was the format that the data was provided in a good format? why or why not?
- 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 manualy (any text editor, including jupyter can edit a `.csv`), making notes about what changes you made in a markdown cell.

> **Think Ahead**
>
> 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 eg this google scholar result How could something like this impact your work as a data scientist?

> **Warning**
>
> This section is not required, but is intended to help you get started thinking about ideas for your portflio. 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.

3. Assignment 3: Exploratory Data Analysis

__Due:2020-09-28 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.

task	skill (max level)
compute and display overall statistics	summarize (2)
compute and display individual statistics of a datasets	summarize (2)
group a data set by a variable and compute summary statics	summarize (2)
plot two different pairwise relationships	visualize (2)
interpret the statistics and plots	summarize (2), visualize (2)
load data from at least two different file types	access (2)
compare and contrast the file types and/or sources	access (2)
match questions appropriate to the dataset and match plots and stats	process (1)

Table 3.1 plot basic views of data and generate descriptive statistics and basic plots

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 using statistics, split-apply-combine, and visualizations. Make a heading for each question using a markdown cell and H2:`##`. Make sure your analyses meet the criteria in the check lists below. Interpret the answer from the analysis/plot.
5. Describe what, if anything might need to be done to clean or prepare this data for further analysis

3.3.1. Dataset 1 Checklist

1. Overall summary statistics grouped by a categorical variable
2. A single statistic grouped by a categorical variable
3. a scatter plot with the points colored by a categorical variable
4. a plot and summary table that convey the same information. This can be one statistic or many.

3.3.2. Dataset 2 Analysis

1. two individual summary statistics for one variable
2. one summary statistic grouped by two categorical variables
3. a figure with a grid of subplots that correspond to two categorical variables
4. 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.

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 portflio. 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.

4. Assignment 4:

Due: 2020-10-05 11:59pm

accept the assignment

task	skill (max level)
drop nan rows from a dataset	prepare (2)
display parts of a dataframe	summarize (1)
impute a value to fill missing values	prepare (2)
filter data based on extreme values or other outliers	prepare (2)
convert a variable to one hot encoding	prepare (2)
add a new column computed from one or more other columns	prepare (2)
transform a dataset to tidy format	prepare (2)
compute overall and individual summary statistics	summarize (2)
use split-apply-combine paradigm	summarize (2)
generate at least two types of plots	visualize (2)
interpret statistics and plots	summarize, visualize
use list comprehensions or loops and pythonic conventions	python (2)
load data from at least two types	access (2)
compare data storage formats	access (2)
match EDA techniques to questions appropriately	process (1)

Table 4.1 practice basic pandas by reshaping and organizing data

For this assignment, prepare the provided datasets. Your preparation needs to include the following steps and narrative description of how you're making decisions about your data cleaning.

The notebooks in the template have instructions for how to work with each dataset.

To earn prepare level 2, clean the data and do just enough exploratory data analysis to show that the data is usable (eg 1 stat and/or plot). For prepare level 2:

- travel_times AND one of:
- cs_degrees, airlines, and coffee

To earn summarize and visualize level 2, add extra exploratory data analyses meeting the criteria above.

To earn python level 2, make sure that you use a function or lambda and comprehension or pythonic loops somewhere. The CS degrees data will have that, but it's harder. The coffee data will be the easiest one to get all python level 2.

For access level 2 you must clean the airline data (to get data in a second file type).

> **Hint**
>
> renaming thing is often done well with a dictionary comprehension or lambda.

Portfolio Dates and Key Facts

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

Current: Check 1

The first portfolio check will be due October 15 and will cover the following skills.

	Level 3	P1	P2	P3	P4
keyword					
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	0
access	access data from both common and uncommon formats and identify best practices for formats in different contexts	1	1	0	0
construct	merge data that is not automatically aligned	1	1	0	0
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	0
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters	1	1	0	0
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	0

Upcoming Checks

Check 2: November 12 Check 3: December 5 Check 4: December 20

Submission Introductions

Your portfolio will be assessed both on your demonstration of skills through your chapters that are inspired by the prompts and that your summary demonstrates that you *know* you learned the skills.

Each portfolio submission, you will edit the corresponding `submission_x_intro.md` file. This is where you write your summary and reflect on your learning. This reflection does not need to be long, it shouldn't take a very long time. It's okay for it to be brief, mostly bullet points.

KWL

One part of your introduction is a **KWL** or **K**now, **W**ant to know, **L**earned, chart.

In your file it will look like this:(but longer)

```
```{list-table} Portfolio 1 KWL Chart
:header-rows: 1
:name: kwl1

* - Skill
  - Know
  - Want to Know
  - Learned
* - python
  - basics
  - more efficient types
  -
* - access
  - that datasets are collated on kaggle
  - how to load data for analysis
  - how to load data from 3 different types and compare them
* - ...
  - ...
  - ...
  - ...
```
```

and when you build your portfolio it will render like this:

Skill	Know	Want to Know	Learned
python	basics	more efficient patterns	pep8 patterns and common conventions,
access	that datasets are collated on kaggle	how to load data for analysis	how to load data from 3 different types and compare them
…	…	…	…

Table 1 Portfolio 1 KWL Chart

Overview

In the overview, you summarize the contents of your portfolio. Think of it as the the introduction to the overall submission. Your goal is to help us know what to expect when grading your portfolio and to know that you know what you've learned.

Writing this out will help give you a space to confirm that you're on track by checking your own work against the Achievement Definitions table. If your work does not earn level 3 achievements, your summary will help us identify if you are on track or if you're not on track. If you're not on track it will help us distinguish between if it's because of a misunderstanding in the expectations or the material.

This summary helps us help you achieve your own goals and lets us help you accordingly. We want you succeed in the course and the best way to do that is to check in frequently.

> 🧠 **Learning Tip**
>
> This reflection process also help you learn better, in addition to being an accountability check. What you draw your attention to gets reinforced in your memory, so reflecting on what you've learned helps you learn better.

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.

Portfolio Check 1 Ideas

Remember you'll be graded against the rubric, but these are ideas for the structure.

Long single analysis

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

This would be one long notebook that covers all of the skills.

Several shorter reflections/analyses

You could also submit a few shorter pieces that in total cover all of the skills. Some example formats:

Tutorial

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

Cheatsheet

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

Blog post

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

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

Correction & Reflection

If you had trouble with an assignment so far, you can revise what you submitted and resubmit it, with reflections and explanation of what you were confused about, what you tried initially, how you eventually figured it out, and explains the correct answer. Then go a little deeper in exploring the topic in that context to also earn level 3.

Practice Problems and Solutions

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

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

- 52 (correct)
- 412 (didn't carry the one, correctly: 7+5 = 12, 3+1 = 4)
- 42 (dropped the one 7+5 = 12, ones place is 2, 3+1 = 4)
- 43 (carried one into wrong column, 7 + 5 = 12, 1+2 = 3, 3+1 = 3)

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 FAQ

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

Can I submit this assignment late if …?

Git and GitHub

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

My command line says I cannot use a password

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

My portfolio won't compile

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

Code Errors

Key Error

<bound method

Glossary

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

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

anonymous function
: a function that's defined on the fly, typically to lighten syntax or return a function within a function. In python, they're defined with the lambda keyword.

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

dictionary
: a mapping array that matches keys to values.

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

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

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

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
- 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.

Requires 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

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.

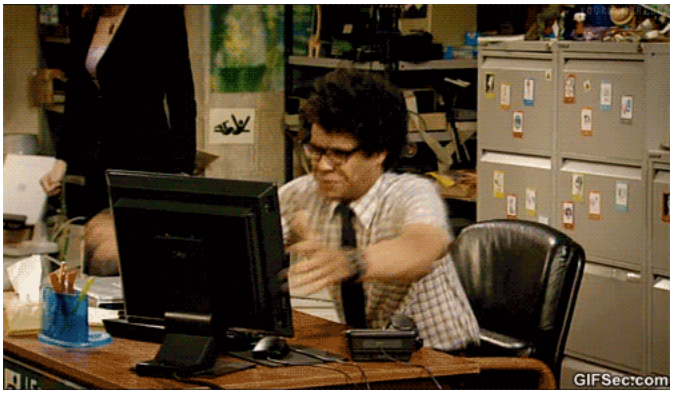
on email

- 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

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.

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.

> **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.
