

Welcome to Programming for Data Science

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

This class meets 3-3:50pm MWF 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

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 [repository](#). You can view the date of changes and exactly what changes were made on the Github [commits](#) page.

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

About your instructor

Name: Dr. Sarah 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 assignment help

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

Using issues

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

For E-mail

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

Note

Whether you use CSC or DSP does not matter.

Tools and Resources

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

All of the tools and resources below are either:

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

BrightSpace

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

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

For announcements, you can [customize](#) how you receive them.

Important

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

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

Prismia chat

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

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

Note

Seeing the BrightSpace site requires logging 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, you will need to [create a Personal Access Token](#) for each device you use.

Programming Environment

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

Requirements:

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

Warning

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

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`

Optional:

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

Video install instructions for Anaconda:

- [Windows](#)
- [Mac](#)
- I don't have a video for linux, but it's a little more straight forward.

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

Textbook

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

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

Zoom (backup 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 mutple 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.

	topics	skills
week		
1	[admin, python review]	process
2	Loading data, Python review	[access, prepare, summarize]
3	Exploratory Data Analysis	[summarize, visualize]
4	Data Cleaning	[prepare, summarize, visualize]
5	Databases, Merging DataFrames	[access, construct, summarize]
6	Modeling, 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 [BrightSpace calendar](#) 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.

	skill	Level 1	Level 2	Level 3
keyword				
python	pythonic code writing	python code that mostly runs, occasional pep8 adherence	python code that reliably runs, frequent pep8 adherence	reliable, efficient, pythonic code that consistently adheres to pep8
process	describe data science as a process	Identify basic components of data science	Describe and define each stage of the data science process	Compare different ways that data science can facilitate decision making
access	access data in multiple formats	load data from at least one format; identify the most common data formats	Load data for processing from the most common formats; Compare and contrast most common formats	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	construct datasets from multiple sources	identify what should happen to merge datasets or when they can be merged	apply basic merges	merge data that is not automatically aligned
summarize	Summarize and describe data	Describe the shape and structure of a dataset in basic terms	compute summary standard statistics of a whole dataset and grouped data	Compute and interpret various summary statistics of subsets of data
visualize	Visualize data	identify plot types, generate basic plots from pandas	generate multiple plot types with complete labeling with pandas and seaborn	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	prepare data for analysis	identify if data is or is not ready for analysis, potential problems with data	apply data reshaping, cleaning, and filtering as directed	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
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 regularized 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 multiple quantitative 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

keyword	skill	Level 1	Level 2	Level 3
unstructured	model	Identify options for representing text data and use them once data is transformed	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
	unstructured data			
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well structured problems with a single tool pipeline	Solve semi-structured, 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 weaknesses of common tools

```

/opt/hostedtoolcache/Python/3.7.11/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: Using the level keyword in DataFrame
and Series aggregations is deprecated and will be removed in a future version. Use
groupby instead. df.sum(level=1) should use df.groupby(level=1).sum().
"""Entry point for launching an IPython kernel.
/opt/hostedtoolcache/Python/3.7.11/x64/lib/python3.7/site-
packages/ipykernel_launcher.py:6: FutureWarning: Using the level keyword in DataFrame
and Series aggregations is deprecated and will be removed in a future version. Use
groupby instead. df.sum(level=1) should use df.groupby(level=1).sum().

```

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	Assignment
keyword														
python	1	1	1	1	0	0	0	0	0	0	0	0	0	
process	1	1	0	0	0	0	0	0	0	0	0	0	0	
access	0	1	1	1	0	0	0	0	0	0	0	0	0	
construct	0	0	0	0	1	1	0	0	0	0	0	0	0	
summarize	0	0	1	1	1	1	1	1	1	1	1	1	1	1
visualize	0	0	1	1	0	1	1	1	1	1	1	1	1	1
prepare	0	0	0	1	1	0	0	0	0	0	0	0	0	
classification	0	0	0	0	0	1	1	0	0	1	0	0	0	
regression	0	0	0	0	0	0	0	1	0	0	1	0	0	
clustering	0	0	0	0	0	0	0	0	1	0	1	0	0	
evaluate	0	0	0	0	0	0	0	0	0	1	1	0	0	
optimize	0	0	0	0	0	0	0	0	0	1	1	0	0	
compare	0	0	0	0	0	0	0	0	0	0	1	0	1	
unstructured	0	0	0	0	0	0	0	0	0	0	0	1	1	
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	

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	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
classification	fit and apply classification models and select appropriate classification models for different contexts		0	1	1	0
regression	can fit and explain regularized 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 quantiatieve 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 weaknesses of common tools		0	0	1	1

Grading

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

Principles of Grading

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

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

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

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

How it works

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

These map onto letter grades roughly as follows:

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

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

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

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

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

Participation

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

Assignments

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

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

Portfolio Checks

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

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

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

TLDR

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

Detailed mechanics

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.

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

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

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

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

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-'
```

Note

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

Late work

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

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

In this issue, include:

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.

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.

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](#) 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

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/bri. 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.

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
-

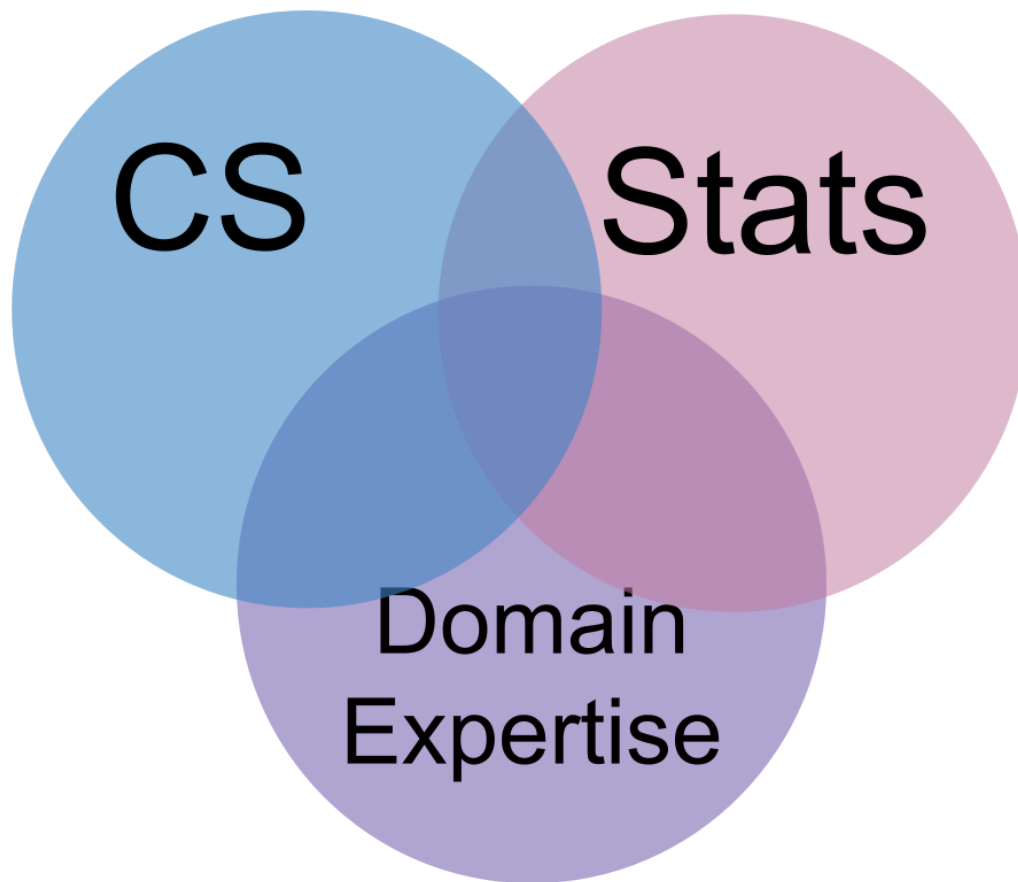
Prismia Chat

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

What is Data Science

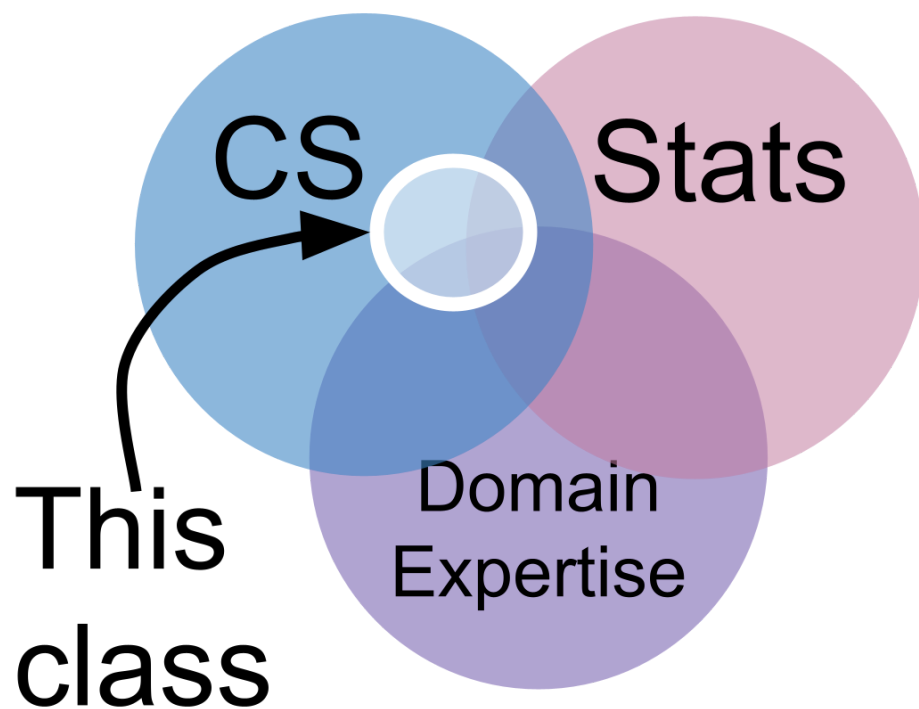
In general:



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.

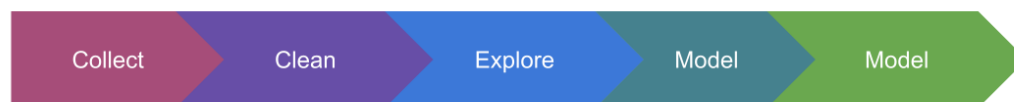
domain expertise helps us have the intuition to know if what we did worked right. A statistic must be interpreted in context; the relevant context determines what they mean and which are valid. The context will say whether automating something is safe or not, it can help us tell whether our code actually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



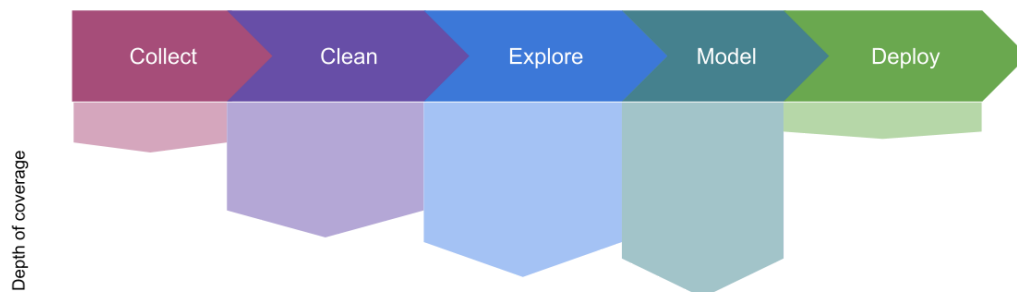
We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

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

How does data science happen?

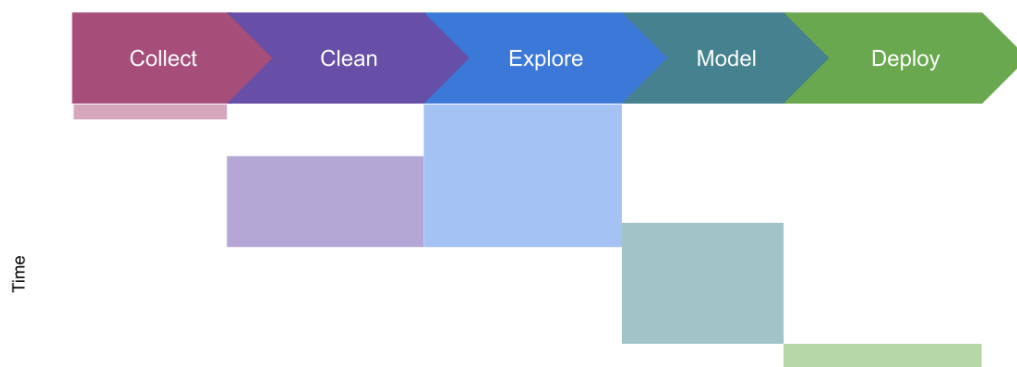


how we'll cover it, in depth



- *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

how we'll cover it in, time



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

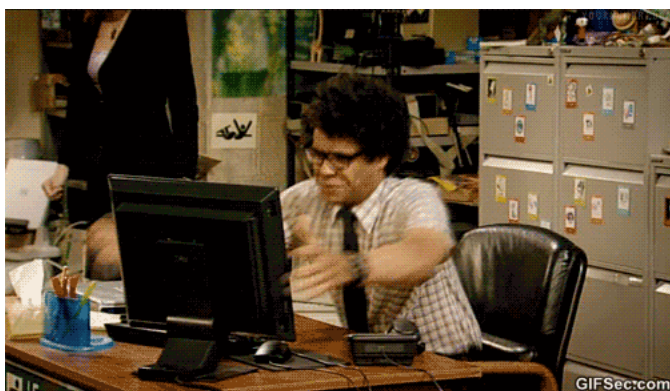
How this class will work

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

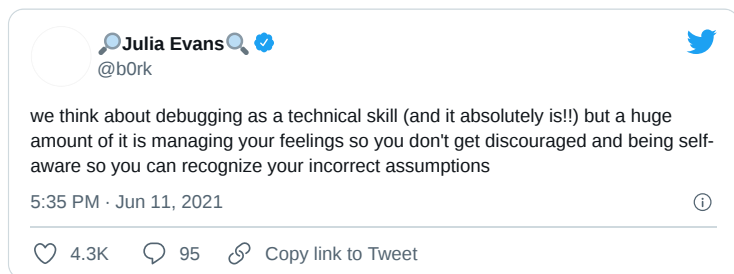
Let's look at the [syllabus](#)

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

Class is designed to avoid this:



Learning Cycle



Read more about how I'm designing this course to help you learn on the [how to learn](#) page.

Jupyter Notebook Tour & Python Review

A jupyter notebook tour

Launch a [jupyter notebook](#):

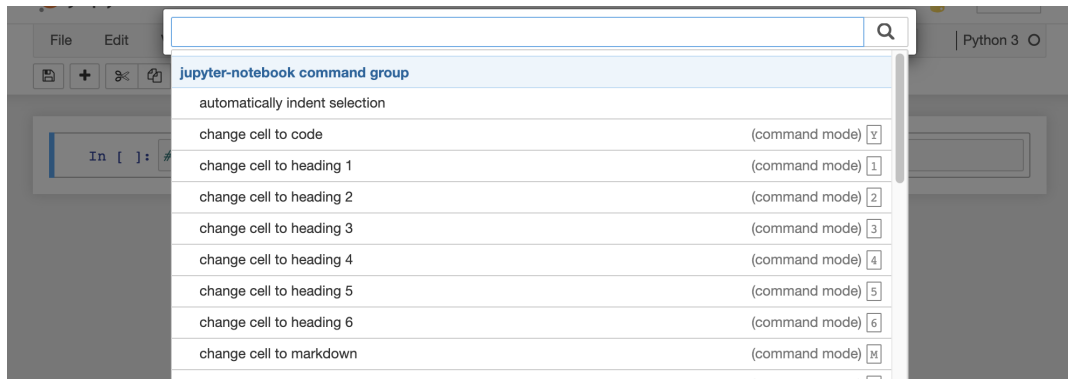
- on Windows, use anaconda terminal
- on Mac/Linux, use terminal

```
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.

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

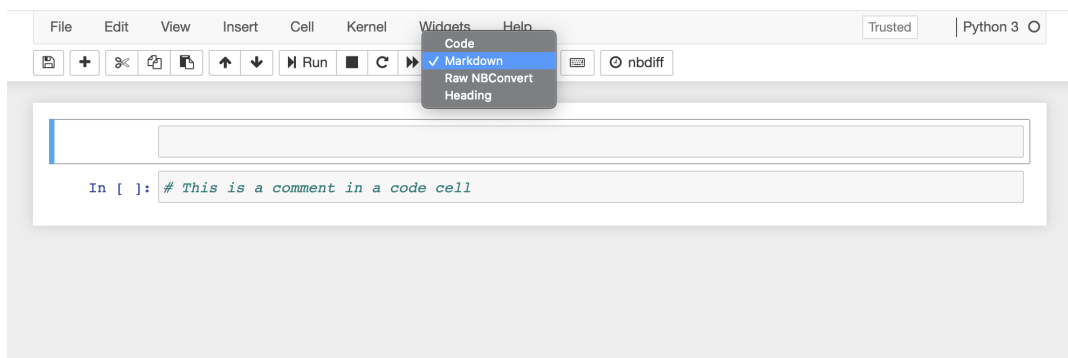
When you press a key in command mode it works like a shortcut. For example **p** shows the command search menu.



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

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

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



++

This is a markdown cell

- we can make
 - itemized lists of
 - bullet points
1. and we can make nubmered
 2. lists, and not have to worry

3. about renumbering them
4. if we add a step in the middle later

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'
```

Just enough Git for Assignment 1

For a little more python review, see my [2020 CSC310 notes](#) this is just enough for this assignment.

Assignment 1:

Goals for this assignment

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

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:

"FINAL".doc



FINAL.doc!



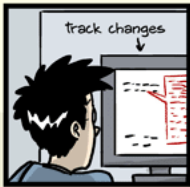
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc

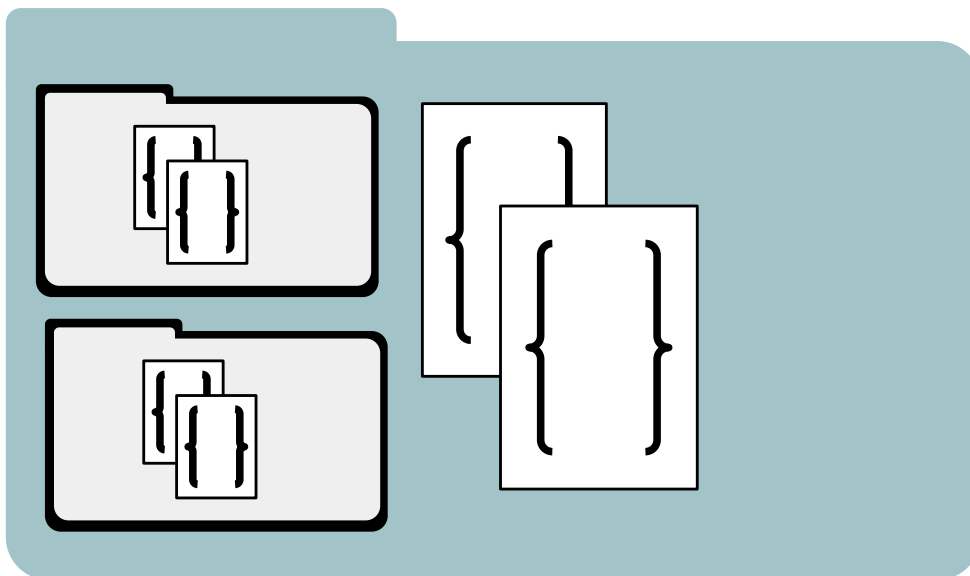


FINAL_rev.22.comments49.
corrections.10. #@\$%WHYDID
ICOMETOGRADSCHOOL????.doc

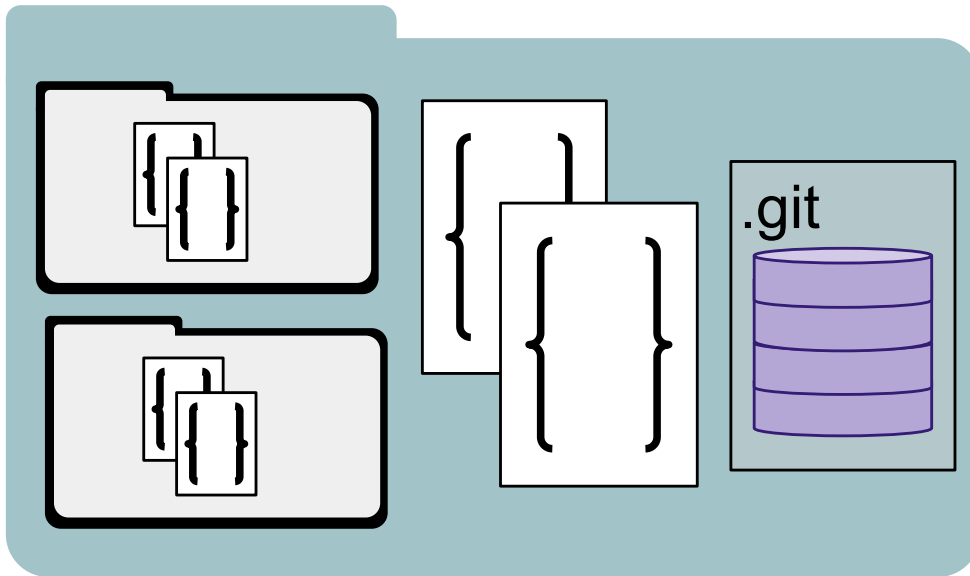
WWW.PHDCOMICS.COM

JORGE CHAM © 2012

We typically organize projects in folder

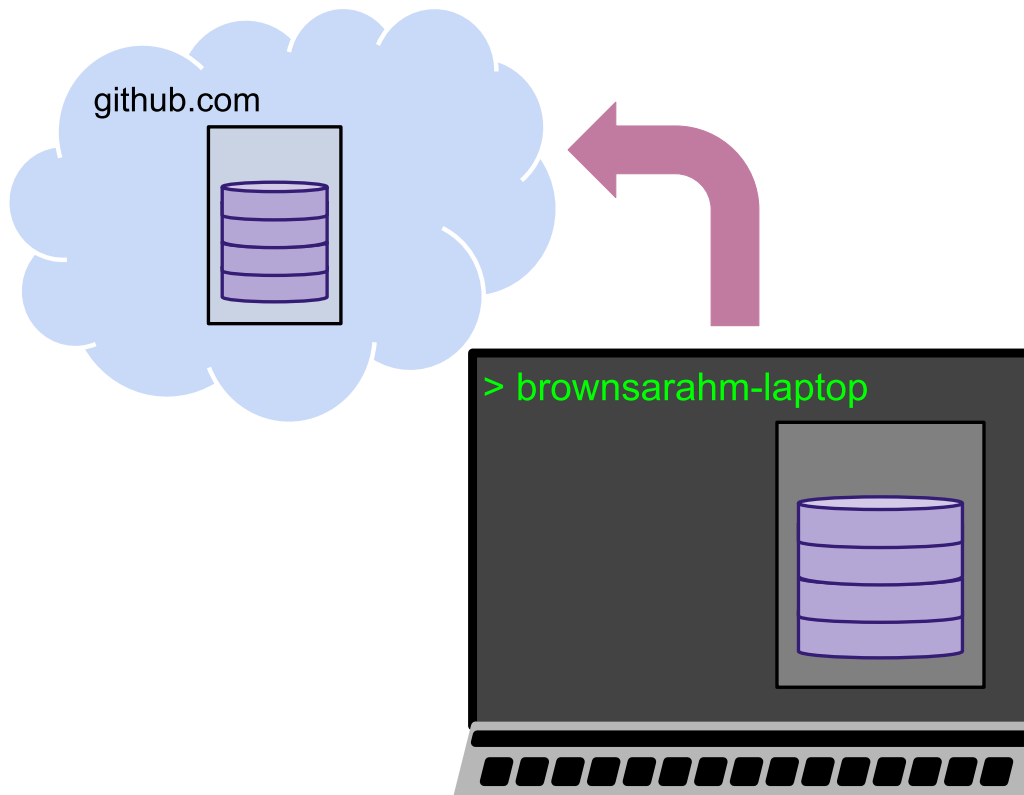


A repository is a folder with a hidden directory named `.git`




The `git` application 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 button that is below the menu area to the right. It's at the top right corner of the list of files in the repository.

 a screen shot of the GitHub

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

```
git pull
```

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

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.


Assignment 1: Portfolio Setup, Data Science, and Python

Due:

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.

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 runnable in the notebook as is.

To Do

Important

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

Your task is to:

1. Install required software from the Tools & Resource page
2. Create your portfolio, by
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, os use anaconda prompt on Windows) 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+'
```

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 with `@rhodyprog4ds/{{ ghinstructors }}`.

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

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.

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 [@brownsarahm](#).

Note

when the value of the expression after `assert` is `True`, it will look like nothing happened. `assert` is used for testing

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 FAQ

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



Common Debugging Issues

Key Error



<bound method



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.

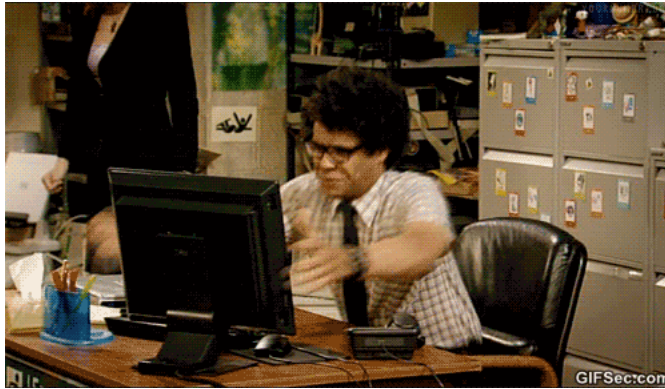
on email

- [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 programming and using data to understand the world. Learning two things at once is more complex. In this page, I break down how I expect learning to work for this class.

Remember the goal is to avoid this:



Why this way?

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

Using reference materials frequently is a built in part of programming, most languages have built in help as a part of the language for this reason.

A new book that might be of interest if you find programming classes hard is [the Programmers Brain](#) As of 2020-09-07, it is available for free by clicking on chapters at that linked table of contents section.

Where are your help tools?

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

Learning in class

Important

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

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

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

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

After class

After class, you should practice with the concepts introduced.

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

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

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

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

There will be additional drills posted to Prismia chat for you to try.

Assignments


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

Portfolios

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

Getting Help with Programming

Asking Questions

 comic on asking questions, that summarizes blog post

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

Describing what you have so far

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

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

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

Understanding Errors

Error messages from the compiler are not always straight forward.

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.

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.

References on Python

- [Course Text](#)

Data Sources

- [UCI Data Repository](#)
- [Json Datasets](#)
- [Databases](#)

- [Google Dataset Search](#)
- [Kaggle](#)

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](#)).

By Professor Sarah M Brown
© Copyright 2021.