Welcome to Programming for Data Science

Welcome to the course manual for CSC310 at URI.

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

Syllabus

Welcome to CSC/DSP310: Programming For Data Science.

In this syllabus you will find an overview of the course, information about your instructor, course policies, restatements of URI policies, reminders of relevant resources, and a schedule for the course.

About

About this course

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

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

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

About this semester

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

About this syllabus

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

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

About your instructor

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

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

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

Note

Whether you use CSC or DSP does not matter.

Tools and Resources

We will use a variety of tools to conduct class and to facilitate your programming. You will need a computer with Linux, MacOS, or Windows. It is unlikely that a tablet or Chromebook will be able to do all of the things required in this course.

All of the tools 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.

Zoom

This is where we will meet for synchronous class sessions. 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.

Important

TL;DR [1]

- check Brightspace
- Install Zoom
- Setup your URI Zoom Account
- 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.

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/BrownFall20/. 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. There will be a link to our class GitHub Classroom on Brightspace.

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, etc)
- Git
- A web browser compatible with Jupyter Notebooks

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.

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.

Textbook

The text for this class is a reference book and iwll 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:

Note

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.

---

[1] Too long; didn't read.

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 is intended to be easier than typical grading. I expect everyone to get at least a C.
- Earning a B in this class is intended to be very accessible, you can make a lot of mistakes along the way as you learn, as long as you learn by the end.
- Earning an A in this class will be challenging, but is possible even with making mistakes while you learn.

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.

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.

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 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.

The letter grade can be computed as follows

Note

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

```
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
```

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

```
'A'
```

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

```
'C-'
```

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

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

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

Late work

No late work will be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally *may* not 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.

Note

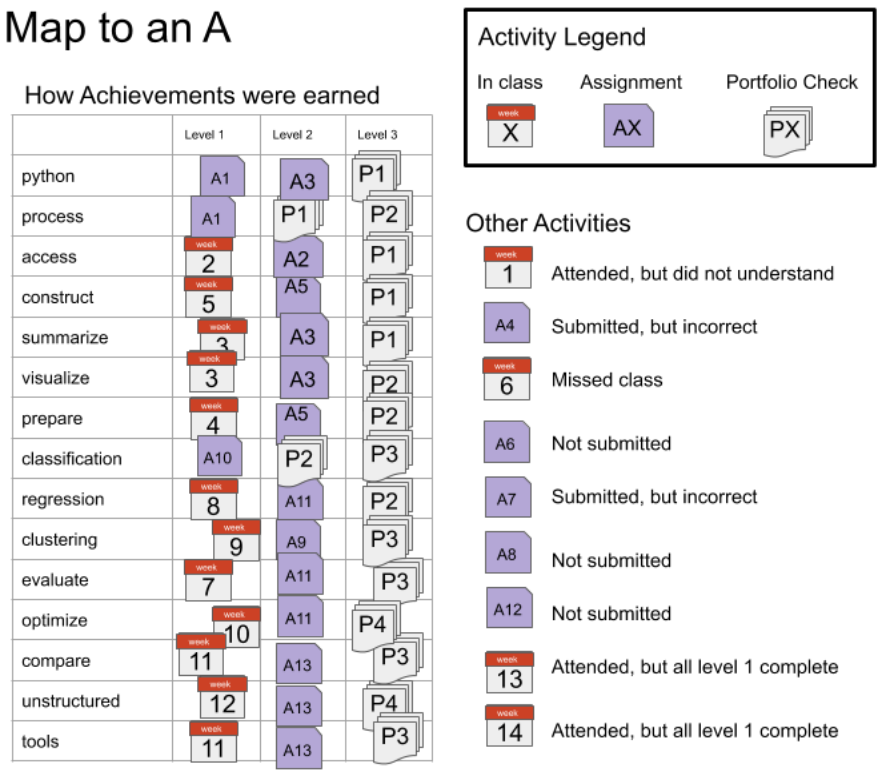
You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Examples

> **Important**
>
> The following will make more sense after you read the next section of the syllabus and see the skills rubric sections.

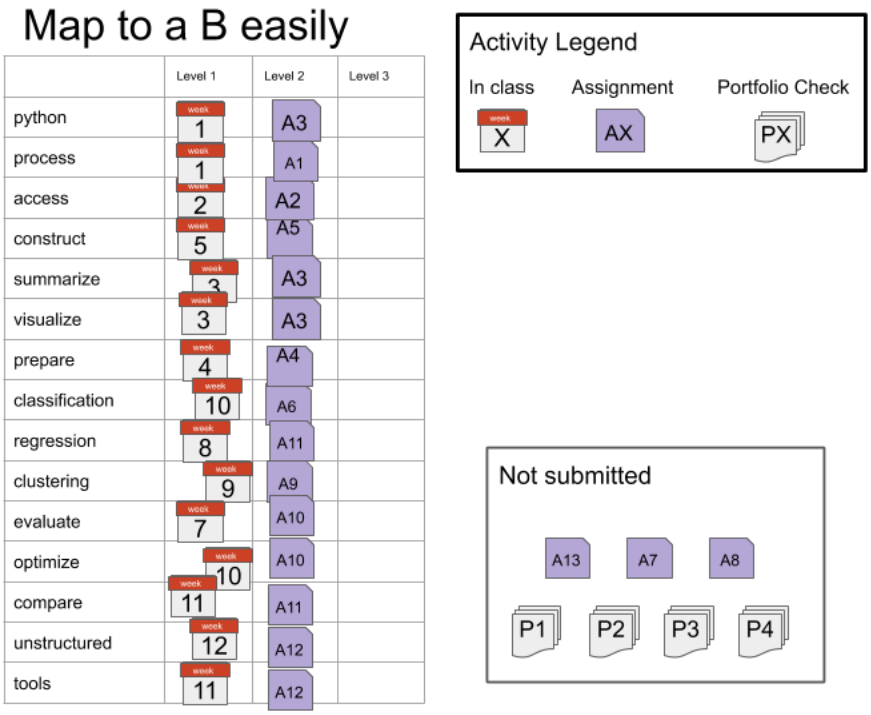
If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check or assignment 13.

Getting A Without Perfection



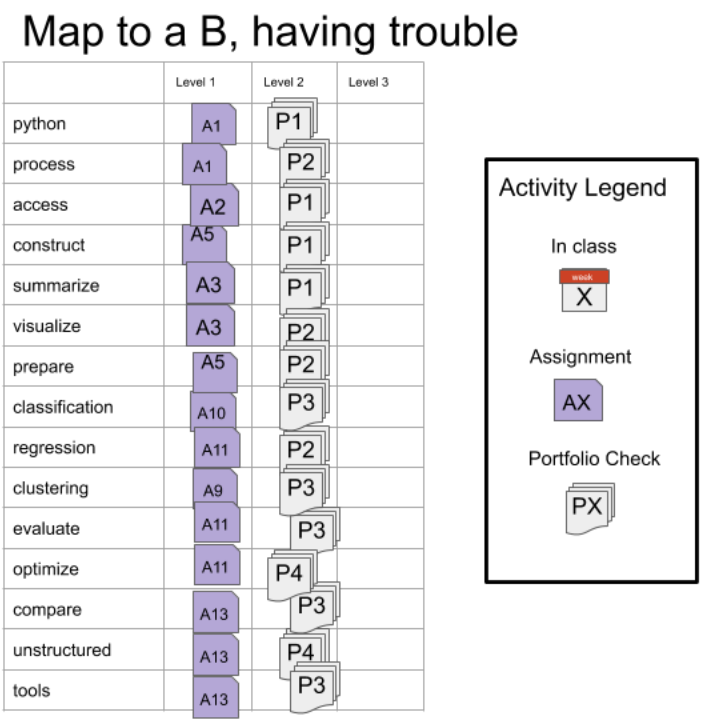
In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level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

Learning Objective, Schedule, and Rubric

Learning Outcomes

There are five learning outcomes for this course.

```
keyword
process        Describe the process of data science, define each phase, and identify
standard tools
data                                    Access and combine data in multiple
formats for analysis
exploratory    Perform exploratory data analyses including descriptive statistics and
visualization
modeling       Select models for data by applying and evaluating mutiple models to a
single dataset
communicate                 Communicate solutions to problems with data in common
industry formats
Name: outcome, dtype: object
```

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 1-1:50pm on Zoom. Every class will include participatory live coding (instructor types, students follow along)) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

> **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.

Programming assignments that will be due each week Sunday 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	Topic Modeling	[unstructured, tools]
14	Deep Learning	[tools, compare]

Skill Rubric

The skill rubric 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 stge of the data science process	Compare different ways that data science can occur
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 statistics of a whole dataset	Compute 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 customize with matplotlib	generate complex plots with pandas and plotting libraries
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 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

keyword	skill	Level 1	Level 2	Level 3
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 mulitple representations and compare and contrast them for different end results
tools	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 with multiple tools	Scope, choose an appropriate tool pipeline and solve data science problems

Assignments and Skills

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

keyword	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
python	1	1	1	1	0	0	0	0	0	0	0	0	0	4
process	1	1	0	0	0	0	0	0	0	0	0	0	0	2
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	0	0	1	1	1	1	1	1	1	1	9
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
tools	0	0	0	0	0	0	0	0	0	1	1	1	1	4

Portfolios and Skills

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

keyword	Level 3	P1	P2	P3	P4
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	0
process	Compare different ways that data science can occur	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 summary statistics of subsets of data	1	1	0	0
visualize	generate complex plots with pandas and plotting libraries	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 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 mulitple representations and compare and contrast them for different end results	0	0	1	1
tools	Scope, choose an appropriate tool pipeline and solve data science problems	0	0	1	1

Support

Academic Enhancement Center

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 through drop-in centers and small group tutoring. 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. They 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 students. At this uncertain time, those concerns include minimizing the potential spread of COVID-19 within our community. While the university has worked this summer 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. Students are required to comply with Rhode Island state laws, including the Rhode Island Executive Orders related to health and safety, ordinances, regulations, and guidance adopted by the University as it relates to public health crises, such as COVID-19.

An addendum on policies and guidelines concerning your obligations during this crisis has recently been integrated into the Student Handbook. These obligations include:

- Wearing of face masks by all community members when on a URI campus in the presence of others
- Maintaining physical distancing of at least six feet at all times

- Following state rules on the number of individuals allowed in a group gathering
- Completing a daily health self-assessment also available through the Rhody Connect app before coming to campus
- Submitting to COVID-19 testing as the University monitors the health of our community
- Following the University's quarantine and isolation requirements

If you answer yes to any of the questions on the daily health assessment, do not go to campus. YOU MUST STAY HOME/IN YOUR ROOM and notify URI Health Services via phone at 401-874-2246 immediately.

If you are already on campus and start to feel ill, you need to remove yourself from the public and notify URI Health Services via phone immediately at 401-874-2246 and go home/back to your room and self-isolate while you await direction from Health Services.

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

Class Notes

Class notes will get posted here day by day

- 2020-09-11: Jupyter Notebook tour, conditionals, functions

Class 2: intro to notebooks and python

Agenda:

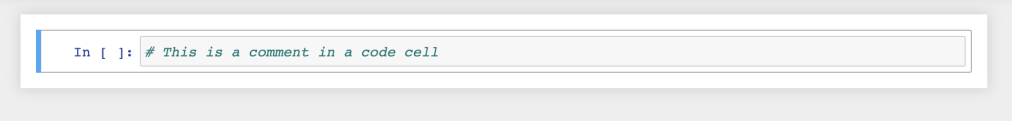
1. review Data Science
2. `jupyter` notebooks
3. `python`: conditionals and functions

Jupyter Notebooks

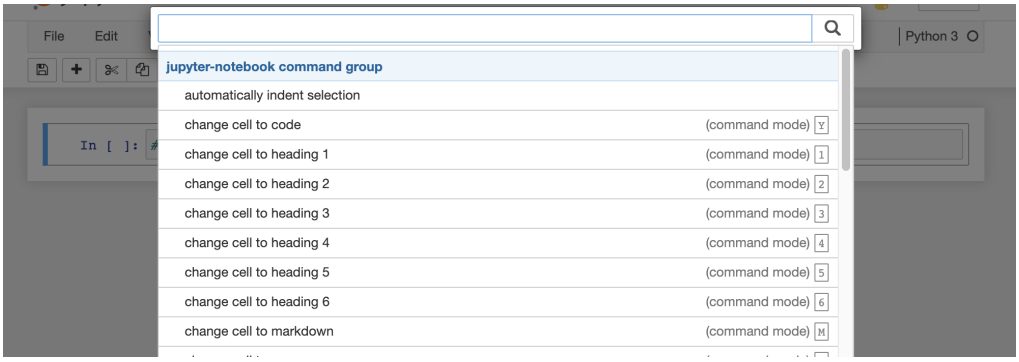
To launch a Jupyter notebook, in your anaconda prompt on Windows or terminal on Linux or Mac:

```
cd dir/you/want/to/work/in
jupyter notebook
```

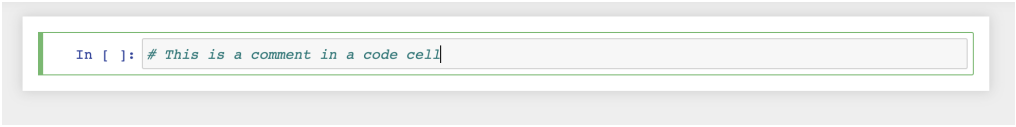
A Jupyter notebook has two modes. When you first open, it is in command mode. The border is blue in command mode.



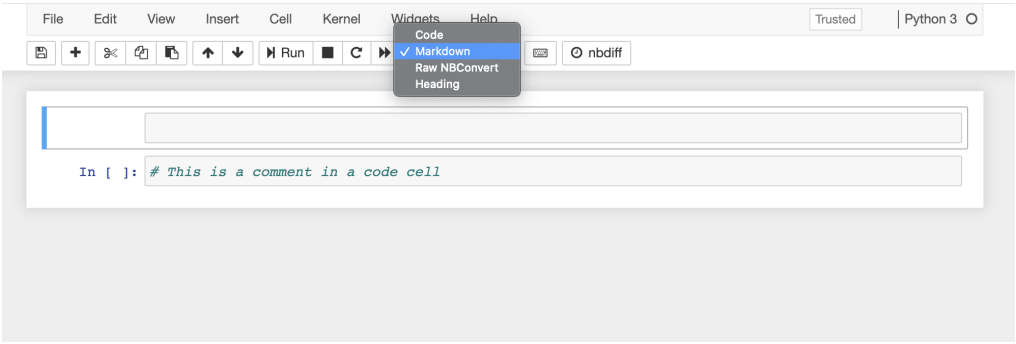
When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.



If you press `enter` (or `return`) or click on the cell it changes to edit mode. The border is green in edit mode



Type code or markdown into boxes called cells. 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.



You can treat markdown cells like plain text, or use special formatting. Here is a [markdown cheatsheet]

Code cells can run like a calculator. If there is a value returned by the last line of a cell, it will be displayed.

```
4+5
```

```
9
```

For example, when we assign, python "returns" `None` so there is no output from this cell

```
a = 9
```

But this one does display the value of the cell

```
a
```

```
9
```

```
b = 4
b
```

```
4
```

```
a
```

```
9
```

Getting Help in Python and Jupyter

The standard way to get help in the help function

```
help(print)
```

Note

Here in class we changed the value of `a` above and noted that the new value is shows here, but not in the previous cell that had ouput the value of `a`

Note that these are green in the jupyter notebook because they're python reserved words.

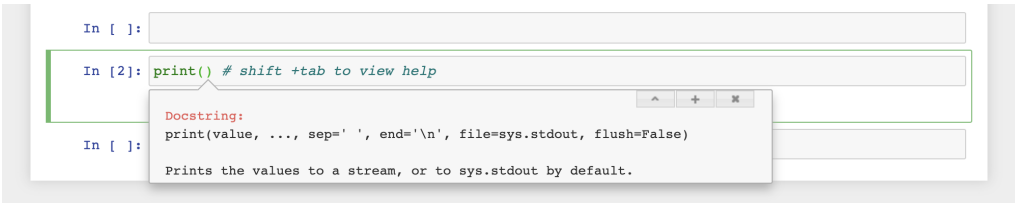
```
Help on built-in function print in module builtins:

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.
```

There are two special ways to get help in Jupyter, one dynamically while you're working and one that stays displayed for a while

Python comments are indicated by the # symbol.

```
print() # shift +tab to view help
```

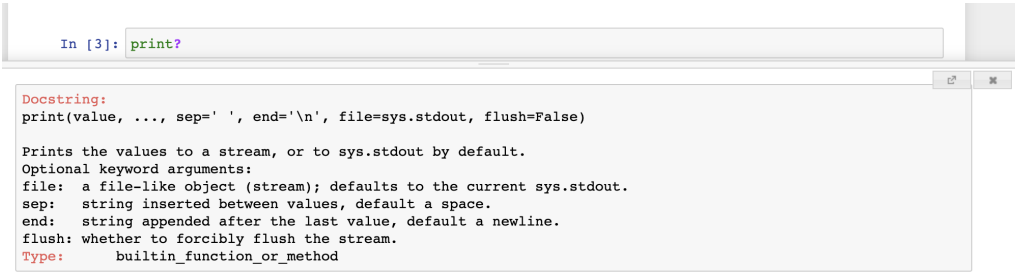
That will look like this:



Press tab twice for the longer version.

A Question mark puts it in a popup window that stays until you close it

```
print?
```



This means you can then use the displayed help to remember how to call the function

```
print(a,b, 'hello',sep='-')
```

```
9-4-hello
```

```
a
b
```

```
4
```

If Statements

> **Warning**
>
> if the *concept* of an `if` in programming is new to you, you should talk to Dr. Brown. Basic programming is a prerequisite to this course, we're *reviewing* basic ideas but only at a level of detail to serve as a reminder.

The synta

```
if a >b:
    print('greater')
```

```
greater
```

```
if b > a:
    print('greater')
```

> **Tip**
>
> this is updated to include things that were skipped in class and discussed after the breakouts

You can check the contents of a string with the `in` keyword

```
name = 'sarah'
if 'a' in name:
    print(name, 'has an a')
```

```
sarah has an a
```

> **Tip**
>
> `in` works for all lists, we'll learn more about that next week

if we copy and change the name we get no output

```
name = 'Beibhinn'
if 'a' in name:
    print(name, 'has an a')
```

Functions

> **Tip**
>
> this is also updated to include things that were skipped in class and discussed after the breakouts

How to write functions in python:

- the `def` keyword starts a function definition
- then the function name
- then the parameters it accepts in `()`
- end that line with a `:`
- the body of the function is spaced over one tab, but Jupyter will do it automatically for you. if it doesn't you might have forgotten the `:`

```
def greeting(name):
    '''
    a function that greets the person name by printing

    parameters
    ----------
    name: string
      a name to be greeeted

    Returns
    -------
    nothing
    '''
    print('hello', name)
```

> **Warning**
>
> this is actually not a great function, because the printing is only a *side effect*. It's better to return the output of the function

```
greeting('sarah')
```

```
hello sarah
```

A better version of that function might be:

```
def greeting(name):
    '''
    a function that greets the person name by printing

    parameters
    ----------
    name: string
      a name to be greeeted

    Returns
    -------
    nothing
    '''
    return 'hello ' + name
```

> **Tip**
>
> you can append strings with +

> **Try it Yourself!**
>
> Write a function that checks if a string has a space in it and returns "please rename" if there is a space.
>
> Remember a docstring. Call your function a couple of times to confirm it works.

Unhide the cell below to see the answer.

This is what some calls of the function look lik

```
check_string("my data.csv")
```

```
'please rename'
```

If there's no string we see no output

```
check_string("my_data.csv")
```

What does python actually return?

```
NoneType
```

Welcome to Week 2

This week we will:

- clarify how this grading really works
- learn about accessing data
- use accessing data as motivation to review more python

Grading and Assignment 1

Iterables

Python has a general data type for objects that are designed to facilitate repetition of some sort, they're called `iterable`s

We've already seen one. Strings are `Iterable`s

```
name = 'sarah'
```

which means we can index them

```
name[3]
```

```
'a'
```

> **Tip**
>
> remember python indexes from 0

Indexing with a negative number counts from the end

```
name[-1]
```

```
'h'
```

Loops in python have similar syntax to the `if` and functions we saw last week:

```
for char in name:
    print(char*3)
```

```
sss
aaa
rrr
aaa
hhh
```

some notes:

- `char` is called the loop variable
- `name` is called the collection- this can be any iterable type object in python
- `print(char*3)` is called the loop body
- python lets us use mathematical operations on strings

Lists and List Comprehensions

We make a list with square brackets

```
names = ['sarah', 'Jose', 'Cam', 'Bri']
```

we can also build lists by folding a loop into the list construction

```
['hello' + n for n in names]
```

```
['hellosarah', 'helloJose', 'helloCam', 'helloBri']
```

this is called a list comprehension

```
greetings = ['hello ' + n for n in names]
```

```
greetings[0]
```

```
'hello sarah'
```

Dictionaries

Dictionaries are a useful datatype in python. It is denoted by `{}` and contains `key: value` pairs separated by commas.

```
gh_names = {'brownsarahm':'Sarah Brown',
            'briannakathrynm1' : 'Brianna MacDonald',
            'jdion62':'Jacob Dion'}
gh_names
```

```
{'brownsarahm': 'Sarah Brown',
 'briannakathrynm1': 'Brianna MacDonald',
 'jdion62': 'Jacob Dion'}
```

You can think of it like a list of the values with a named index.

```
gh_names['jdion62']
```

```
'Jacob Dion'
```

we can iterate over both the key and the value by using the `items` method on a dictionary. That makes another iterable object that can be used as a loop collection. It functions as a set of pairs now, so we get two loop variables:

```
for key, value in gh_names.items():
    print(value, "'s username is ", key)
```

```
Sarah Brown 's username is  brownsarahm
Brianna MacDonald 's username is  briannakathrynm1
Jacob Dion 's username is  jdion62
```

If we iterate over the dictionary without that method, we get the keys.

```
for val in gh_names:
    print(val)
```

```
brownsarahm
briannakathrynm1
jdion62
```

Libraries

To use libraries in python we import them

We will use pandas a lot in this class. It's the Python Data Analysis Library.

```
import pandas
```

Once we import we can use the functions, datatypes, and values a library provides by using a `.` after the name. In a notebook, pressing tab will show you the options.

```
pandas.read_csv()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-374a1a6f9f7e> in <module>
----> 1 pandas.read_csv()

TypeError: read_csv() missing 1 required positional argument: 'filepath_or_buffer'
```

We can also use an alias to give a library a nickname to make it easier to use. `pd` is the standard alias for `pandas`

```
import pandas as pd
```

We can read in from a local path or a url. Let's read in the course map page of our course website.

```
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')
```

This makes a `list` of `pandas.DataFrame` objects. We can check that with the following

> **Warning**
>
> This cell was added after class, but the explanation was given in class

```
type(pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html'
))
```

To work with it though, we should save to a variable, then we can index into that list.

> **Tip**
>
> note that `import` is a keyword and that in a Jupyter notebook, we can import anywhere and then the library can be used in any cell that is *run* after the import cell is *run*. It's good practice to put them at the top and make your notebook runnable in sequence, but Jupyter won't force you to.

> **Tip**
>
> For example if you don't remember what kind of read functions there are in pandas, type `pandas.read` and then press tabb to see options.

```
df_list =
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')
df_list[0]
```

When you display `DataFrames` in jupyter, they get nice formatting.

Today's Review:

- strings are iterable
- loops
- dictionaries
- imported pandas and read data

Assignments

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

Assignment TOC:

Assignment 1: Portfolio Setup, Data Science, and Python

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.

To Do

Your task is to:

1. Install required software
2. Setup your portfolio, by accepting the assignment and following the instructions in the README file on your repository.
3. Add your own definition of data science to the introduction of your portfolio, in `about/index.md`
4. 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.

1. Uncomment the line `#  - file: about/grading` in your `_toc.yml` file.

> **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/fall20instructors`.
>
> To do this click Issues at the top, the green "New Issue" button and then type away.

```
    '''
    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

Submission Instructions

Create a Jupyter Notebook with your function and Add the notebook to your portfolio by uploading it to your repository, or adding to the folder off line and committing and pushing the changes.

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 the team `rhodyprog4ds/Fall20instructors`.

Solutions

One solution is added to the Detailed Mechanics part of the Grading section of the syllabus.

Assignment 2: Practicing Python and Accessing Data

Objective & Evaluation

This assignment is an opportunity to earn level 1 or 2 achievements in `python`, `process` and `access` and begin working toward level 1 in `summarize`.

Accept the assignment on GitHub Classroom. It contains a notebook with some template structure (and will set you up for grading). The template will also convert notebooks that are added to markdown, which makes reading on GitHub for easier grading. If you want to incorporate feedback you receive back into a notebook file, Jupytext can do that.

To work with this notebook you can either:

- download the repository as .zip from the green code button, unzip, and re-upload, OR
- clone the repository with git and the push your changes. See Git/GitHub help on cloning, committing, and pushing, for example this tutorial on git to learn more about git.

Accessing Data with Python and pandas

(for python and access)

Find 3 datasets of interest to you that are provided in 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. Complete a dictionary for each with the url, a name, and what function should be used to load the data into a `pandas.DataFrame`.

Use a list of those dictionaries to iterate over the datasets and build a table that describes them, with the following columns `['name','source','num_rows', 'num_columns','source_file_name']`. The `source_file_name` should be the part of the url after the last `/`. Display that summary table as a dataframe and save it as a csv.

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

For a dataset that includes nonnumerical data:

- display the heading with the last seven rows
- make and display a new data frame with only the non numerical columns
- was the format that the data was provided in a good format? why or why not?

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)

For the third dataset:

- display the first 5 even rows of the data for three columns of your choice

For any dataset:

- try reading it in with the wrong `read_` function. If you had done this by accident, how could you tell?

Data Science Process

(for the `process` skill)

Make a list of a data science pipeline and denote which types of programming might be helpful at each staged.
Include this in a markdown cell in the same notebook with your analysis

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 …?

GitHub FAQ

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