

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 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, 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 it 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](#):

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 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

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.

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.

Examples

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.

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 an A and you won't need to submit the 4th portfolio check or assignment 13.

Getting A Without Perfection

Map to an A

How Achievements were earned

	Level 1	Level 2	Level 3
python	A1	A3	P1
process	A1	P1	P2
access	2	A2	P1
construct	5	A5	P1
summarize	3	A3	P1
visualize	3	A3	P2
prepare	4	A5	P2
classification	A10	P2	P3
regression	8	A11	P2
clustering	9	A9	P3
evaluate	7	A11	P3
optimize	10	A11	P4
compare	11	A13	P3
unstructured	12	A13	P4
tools	11	A13	P3

Activity Legend

In class	Assignment	Portfolio Check
















Other Activities

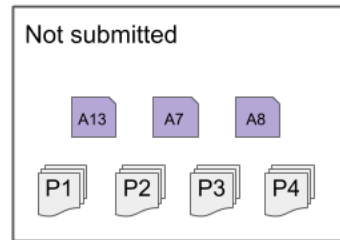
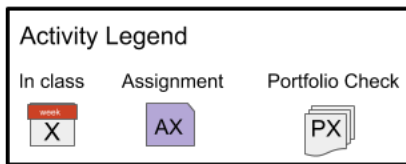
	Attended, but did not understand
	Submitted, but incorrect
	Missed class
	Not submitted
	Submitted, but incorrect
	Not submitted
	Not submitted
	Attended, but all level 1 complete
	Attended, but all level 1 complete

In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the **python**, **process**, and **classification** skills, the level 1 achievements were earned on assignments, not in class. For the **process** and **classification** skills, the level 2 achievements were not earned on assignments, only on portfolio checks, but they were earned on the first portfolio of those skills, so the level 3 achievements were earned on the second portfolio check for that skill. This student's fourth portfolio only demonstrated two skills: **optimize** and **unstructured**. It included only 1 analysis, a text analysis with optimizing the parameters of the model. Assignments 4 and 7 were both submitted, but didn't earn any achievements, the student got feedback though, that they were able to apply in later assignments to earn the achievements. The student missed class week 6 and chose to not submit assignment 6 and use week 7 to catch up. The student had too much work in another class and chose to skip assignment 8. The student tried assignment 12, but didn't finish it on time, so it was not graded, but the student visited office hours to understand and be sure to earn the level 2 **unstructured** achievement on assignment 13.

Getting a B with minimal work

Map to a B easily
















	Level 1	Level 2	Level 3
python	 1	A3	
process	 1	A1	
access	 2	A2	
construct	 5	A5	
summarize	 3	A3	
visualize	 3	A3	
prepare	 4	A4	
classification	 10	A6	
regression	 8	A11	
clustering	 9	A9	
evaluate	 7	A10	
optimize	 10	A10	
compare	 11	A11	
unstructured	 12	A12	
tools	 11	A12	

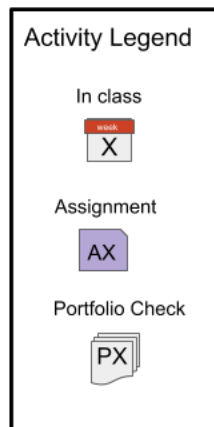


In this example, the student earned all level 1 achievements in class and all level 2 on assignments. This student was content with getting a B and chose to not submit a portfolio.

Getting a B while having trouble

Map to a B, having trouble

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



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

Learning Objective, Schedule, and Rubric

Learning Outcomes

There are five learning outcomes for this course.

1. (process) Describe the process of data science, define each phase, and identify standard tools
2. (data) Access and combine data in multiple formats for analysis
3. (exploratory) Perform exploratory data analyses including descriptive statistics and visualization
4. (modeling) Select models for data by applying and evaluating multiple models to a single dataset
5. (communicate) Communicate solutions to problems with data in common industry formats

We will build your skill in the **process** and **communicate** outcomes over the whole semester. The middle three skills will correspond roughly to the content taught for each of the first three portfolio checks.

Schedule

The course will meet 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.

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

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.

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.

	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 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 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 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

	skill	Level 1	Level 2	Level 3
keyword				
	unstructured	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	model unstructured data			
	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred problems with a single tool pipeline	Solve semi-structured, 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.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
keyword														
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.

		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 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
- [2020-09-14](#): Iterables, Pandas
- [2020-09-16](#): Pandas loading and exploring
- [2020-09-18](#): Pandas, Functions as Object, Dictionaries

Class 2: intro to notebooks and python

Agenda:

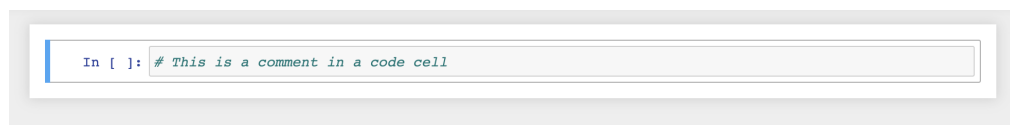
1. review Data Science
2. **jupyter** notebooks
3. **python**: conditionals and functions

Jupyter Notebooks

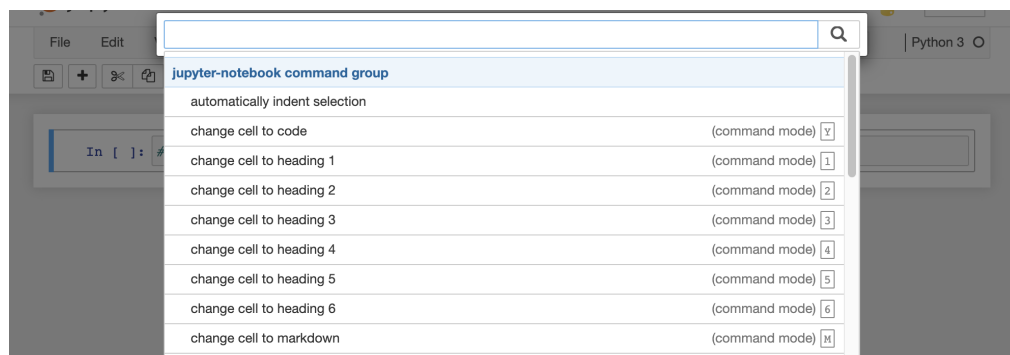
To launch a Jupyter notebook, in your anaconda prompt on Windows or terminal on Linux or Mac:

```
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.



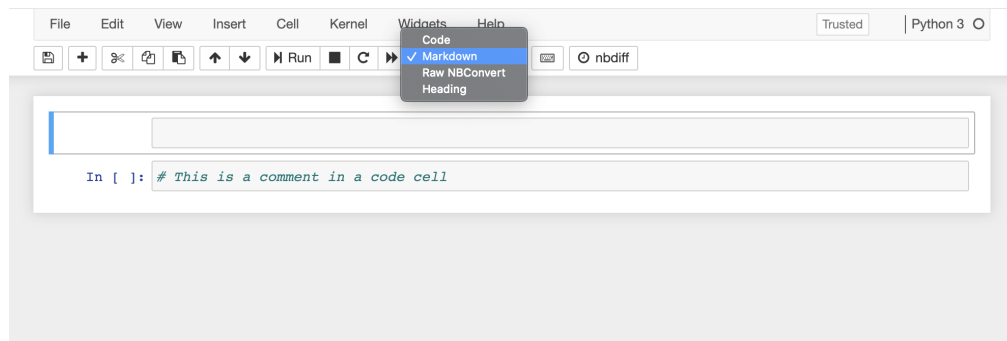
When you press a key in command mode it works like a shortcut. For example **p** shows the command search menu.



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

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

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:

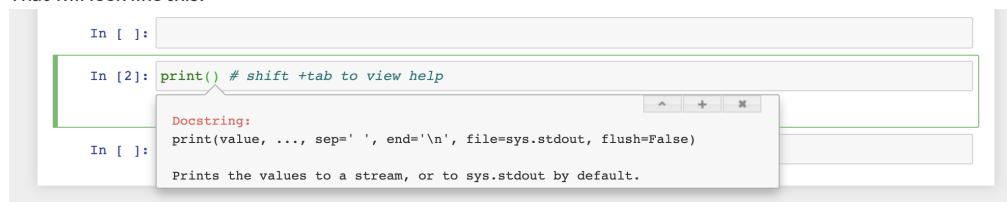
```
print(...)  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)  
  
    Prints the values to a stream, or to sys.stdout by default.  
    Optional keyword arguments:  
    file: a file-like object (stream); defaults to the current sys.stdout.  
    sep:  string inserted between 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.

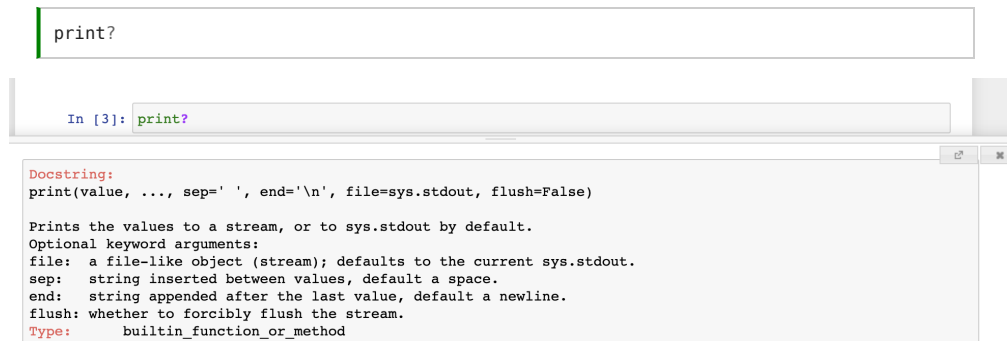
```
print() # shift +tab to view help
```

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



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)
```

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

hello sarah

⚠ 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

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 like

```
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
```

Class 3: 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

- Solution function posted.
- note: not a sum
- read the rubric
- Brightspace
- In class
- Portfolio
 - will start posting prompts The docstring functions like a property of the function object. so it has to be inside.

Iterables

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

We've already seen one. Strings are **Iterables**

```
name = 'sarah'
```

which means we can index them

```
name[3]
```

```
'a'
```

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'
```

Tip

remember python indexes from 0

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',
            'briannakathrynml': 'Brianna MacDonald',
            'jdion62': 'Jacob Dion'}
gh_names
```

```
{'brownsarahm': 'Sarah Brown',
 'briannakathrynml': '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 briannakathrynml
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
briannakathrynml
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')
```

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 tab to see options.

```

-----
ImportError                                Traceback (most recent call last)
<ipython-input-16-a6f048ad97a0> in <module>
----> 1
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    294         )
    295         warnings.warn(msg, FutureWarning, stacklevel=stacklevel)
--> 296         return func(*args, **kwargs)
    297
    298         return wrapper

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in read_html(io, match, flavor, header, index_col, skiprows, attrs, parse_dates,
thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
    1099         na_values=na_values,
    1100         keep_default_na=keep_default_na,
--> 1101         displayed_only=displayed_only,
    1102     )

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parse(flavor, io, match, attrs, encoding, displayed_only, **kwargs)
    892         retained = None
    893         for flav in flavor:
--> 894             parser = _parser_dispatch(flav)
    895             p = parser(io, compiled_match, attrs, encoding, displayed_only)
    896

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parser_dispatch(flavor)
    849         else:
    850             if not _HAS_LXML:
--> 851                 raise ImportError("lxml not found, please install it")
    852             return _valid_parsers[flavor]
    853

ImportError: lxml not found, please install it

```

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'
))

```

```

.....
ImportError                                Traceback (most recent call last)
<ipython-input-17-650e2b275e4e> in <module>
----> 1
type(pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.htm
l'))

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    294         )
    295         warnings.warn(msg, FutureWarning, stacklevel=stacklevel)
--> 296         return func(*args, **kwargs)
    297
    298         return wrapper

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in read_html(io, match, flavor, header, index_col, skiprows, attrs, parse_dates,
thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
    1099     na_values=na_values,
    1100     keep_default_na=keep_default_na,
--> 1101     displayed_only=displayed_only,
    1102 )

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parse(flavor, io, match, attrs, encoding, displayed_only, **kwargs)
    892     retained = None
    893     for flav in flavor:
--> 894         parser = _parser_dispatch(flav)
    895         p = parser(io, compiled_match, attrs, encoding, displayed_only)
    896

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parser_dispatch(flavor)
    849     else:
    850         if not _HAS_LXML:
--> 851             raise ImportError("lxml not found, please install it")
    852     return _valid_parsers[flavor]
    853

ImportError: lxml not found, please install it

```

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

```

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

```

```

-----
ImportError                                Traceback (most recent call last)
<ipython-input-18-16f4763c8ff6> in <module>
----> 1 df_list =
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')
   2 df_list[0]

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
   294         )
   295         warnings.warn(msg, FutureWarning, stacklevel=stacklevel)
--> 296         return func(*args, **kwargs)
   297
   298         return wrapper

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in read_html(io, match, flavor, header, index_col, skiprows, attrs, parse_dates,
thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
   1099     na_values=na_values,
   1100     keep_default_na=keep_default_na,
-> 1101     displayed_only=displayed_only,
   1102 )

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parse(flavor, io, match, attrs, encoding, displayed_only, **kwargs)
   892     retained = None
   893     for flav in flavor:
--> 894         parser = _parser_dispatch(flav)
   895         p = parser(io, compiled_match, attrs, encoding, displayed_only)
   896

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parser_dispatch(flavor)
   849     else:
   850         if not _HAS_LXML:
--> 851             raise ImportError("lxml not found, please install it")
   852         return _valid_parsers[flavor]
   853

ImportError: lxml not found, please install it

```

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

Today's Review:

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

Class 4: Pandas

Today we will:

Remember, Programming is a Practice

- if you're curious about something try it
- you don't need me to give you answers about how code works, the interpreter will tell you
- if you don't remember details, remember you can get help from Jupyter

with a `?` after the function name withouth `()`

```
print?
```

or using the `tab` key inside the `()` for a function

```
print()
```

or from the core python, with the `help` fucntion

```
help(print)
```

Help on built-in function print in module builtins:

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

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

Data in Pandas

We can import `pandas` again as before

```
import pandas as pd
```

and we can read in data.

```
pd.read_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_clean.csv')
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	

to be able to use this, we need to save it to a variable.

```
safi_df = pd.read_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_clean.csv')
```

This is an excerpt from the [SAFI dataset](#).

Another important thing to do is to check datatypes, this is how we know what things we can do with a variable.

```
type(safi_df)
```

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

An important thing to check is the size of the dataset.

```
safi_df.shape
```

```
(131, 14)
```

Recall that you can also tab complete

```
safi_df.shape
```

```
(131, 14)
```

Tip

we can also see the size when we print the whole dataset as in the first time we read the data in.

To see the first 5 rows of the dataset, use the `head()` function

```
safi_df.head()
```

	key_ID	village	interview_date	no_memb	years_liv	respondent_wall_type	rooms	memb_
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11-17T00:00:00Z	7	40	burntbricks	1	

We can call this function with a value to change how many rows are returned

```
safi_df.head(3)
```

	key_ID	village	interview_date	no_memb	years_liv	respondent_wall_type	rooms	memb_
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	

To know how this works, we can view the documentation for the function

```
help(safi_df.head)
```

Warning

this was changed from using `?` for help in class so that the help is displayed in the rendered website, but the pop up was fine in real time

Help on method head in module pandas.core.generic:

head(n: int = 5) -> ~FrameOrSeries method of pandas.core.frame.DataFrame instance
Return the first `n` rows.

This function returns the first `n` rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.

For negative values of `n`, this function returns all rows except the last `n` rows, equivalent to `df[:-n]`.

Parameters

n : int, default 5
Number of rows to select.

Returns

same type as caller
The first `n` rows of the caller object.

See Also

DataFrame.tail: Returns the last `n` rows.

Examples

```
>>> df = pd.DataFrame({'animal': ['alligator', 'bee', 'falcon', 'lion',  
...                               'monkey', 'parrot', 'shark', 'whale', 'zebra']})  
>>> df
```

```
   animal  
0  alligator  
1      bee  
2    falcon  
3      lion  
4    monkey  
5    parrot  
6     shark  
7     whale  
8     zebra
```

Viewing the first 5 lines

```
>>> df.head()  
   animal  
0  alligator  
1      bee  
2    falcon  
3      lion  
4    monkey
```

Viewing the first `n` lines (three in this case)

```
>>> df.head(3)  
   animal  
0  alligator  
1      bee  
2    falcon
```

For negative values of `n`

```
>>> df.head(-3)  
   animal  
0  alligator  
1      bee  
2    falcon  
3      lion  
4    monkey  
5    parrot
```

Since it says `n = 5` we know that the default value of the parameter `n` is 5. When a function has a default value, we can call the function without a value.

To view the last few lines, we use `tail`

```
safi_df.tail()
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
126	126	Ruaca	2017-05-18T00:00:00Z	3	7	burntbricks	1	
127	193	Ruaca	2017-06-04T00:00:00Z	7	10	cement	3	
128	194	Ruaca	2017-06-04T00:00:00Z	4	5	muddaub	1	
129	199	Chirodzo	2017-06-04T00:00:00Z	7	17	burntbricks	2	
130	200	Chirodzo	2017-06-04T00:00:00Z	8	20	burntbricks	2	

We can also get an **Index** for the columns of the DataFrame.

```
safi_df.columns
```

```
Index(['key_ID', 'village', 'interview_date', 'no_membrs', 'years_liv',
      'respondent_wall_type', 'rooms', 'memb_assoc', 'affect_conflicts',
      'liv_count', 'items_owned', 'no_meals', 'months_lack_food',
      'instanceID'],
      dtype='object')
```

[Index](#) is another data type that is defined by **pandas**.

an **Index** variable is iterable so we can index into it

Try it Yourself

How would you view the name of the 3rd column?

First the correct answer:

```
'interview_date'
```

Now some misconceptions:

```
safi_df['interview_date']
```

```
0      2016-11-17T00:00:00Z
1      2016-11-17T00:00:00Z
2      2016-11-17T00:00:00Z
3      2016-11-17T00:00:00Z
4      2016-11-17T00:00:00Z
...
126     2017-05-18T00:00:00Z
127     2017-06-04T00:00:00Z
128     2017-06-04T00:00:00Z
129     2017-06-04T00:00:00Z
130     2017-06-04T00:00:00Z
Name: interview_date, Length: 131, dtype: object
```

Indexing with the column name) will return the *values* in the column

```
safi_df.columns(2)
```

```

-----
TypeError                                Traceback (most recent call last)
<ipython-input-17-bd02c7e8a4a6> in <module>
----> 1 safi_df.columns(2)

TypeError: 'Index' object is not callable

```

Using `()` returns an error, because `columns` is a *property* which is referenced as is with no `()`. We get a type error because functions in python are objects of type `callable` and properties are values not functions.

```
pd.DataFrame.columns[2]
```

```

-----
TypeError                                Traceback (most recent call last)
<ipython-input-18-40e277f3074e> in <module>
----> 1 pd.DataFrame.columns[2]

TypeError: 'pandas._libs.properties.AxisProperty' object is not subscriptable

```

This doesn't work because `columns` is a property of an object of type `pandas.DataFrame` not a property of the `type`

We can see what the type of `pd.DataFrame` is with the `type` function.

```
type(pd.DataFrame)
```

```
type
```

Knowing about types is helpful for the individual columns of a dataset as well.

```
safi_df.dtypes
```

```

key_ID          int64
village         object
interview_date  object
no_membrs       int64
years_liv       int64
respondent_wall_type  object
rooms           int64
memb_assoc      object
affect_conflicts  object
liv_count       int64
items_owned     object
no_meals        int64
months_lack_food  object
instanceID      object
dtype: object

```

Note that it uses `int64` and `object` as the types.

```
safi_df.head(2)
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	

We might want to look at what villages were included in the data.

```
pd.unique(safi_df['village'])
```

```
array(['God', 'Chirodzo', 'Ruaca'], dtype=object)
```

We can also get count of the number of each value

```
safi_df['village'].value_counts()
```

```
Ruaca      49  
God        43  
Chirodzo   39  
Name: village, dtype: int64
```

Try it Yourself!

how many surveyed farms have all type `mauddaub`?

Class 5: Accessing Data, continued

Today's agenda:

- warm up/ review
- announcements
- working with dataframes
- the power of functions as objects
- (maybe) exploratory data analysis

Warm up

To do:

1. start a jupyter notebook
2. check the questin on prismia chat

Read the tables off of the syllabus course map page with `read_html` and make a list of the shapes of all of the tables on the page. Save the output to a variable and paste the *value* of that variable as your answer to the question.

```
import pandas as pd  
[df.shape for df in  
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')]
```

```

-----
ImportError                                Traceback (most recent call last)
<ipython-input-1-013e4411b1bb> in <module>
      1 import pandas as pd
----> 2 [df.shape for df in
pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')]

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    294     )
    295     warnings.warn(msg, FutureWarning, stacklevel=stacklevel)
--> 296     return func(*args, **kwargs)
    297
    298     return wrapper

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in read_html(io, match, flavor, header, index_col, skiprows, attrs, parse_dates,
thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
    1099     na_values=na_values,
    1100     keep_default_na=keep_default_na,
-> 1101     displayed_only=displayed_only,
    1102 )

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parse(flavor, io, match, attrs, encoding, displayed_only, **kwargs)
    892     retained = None
    893     for flav in flavor:
--> 894         parser = _parser_dispatch(flav)
    895         p = parser(io, compiled_match, attrs, encoding, displayed_only)
    896

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parser_dispatch(flavor)
    849     else:
    850         if not _HAS_LXML:
--> 851             raise ImportError("lxml not found, please install it")
    852     return _valid_parsers[flavor]
    853

ImportError: lxml not found, please install it

```

```
safi_df = pd.read_csv('https://raw.githubusercontent.com/brownsarahm/python-socialsci-
files/master/data/SAFI_clean.csv')
```

```
safi_df.to_csv('safi_clean.csv')
```

```
safi_df.to_csv('data/safi_clean.csv')
```

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-4-c8dec1e0d1b9> in <module>
----> 1 safi_df.to_csv('data/safi_clean.csv')

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/generic.py in to_csv(self, path_or_buf, sep, na_rep,
float_format, columns, header, index, index_label, mode, encoding, compression,
quoting, quotechar, line_terminator, chunksize, date_format, doublequote, escapechar,
decimal, errors)
    3165         decimal=decimal,
    3166     )
-> 3167     formatter.save()
    3168
    3169     if path_or_buf is None:

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/formats/csvs.py in save(self)
    188         encoding=self.encoding,
    189         errors=self.errors,
--> 190         compression=dict(self.compression_args,
method=self.compression),
    191     )
    192     close = True

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/common.py
in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors)
    491     if encoding:
    492         # Encoding
--> 493         f = open(path_or_buf, mode, encoding=encoding, errors=errors,
newline="")
    494     elif is_text:
    495         # No explicit encoding

FileNotFoundError: [Errno 2] No such file or directory: 'data/safi_clean.csv'

```

```
safi_df2= pd.read_csv('data/safi_clean.csv')
```

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-5-e632c85491a6> in <module>
----> 1 safi_df2= pd.read_csv('data/safi_clean.csv')

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header,
names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,
converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,
infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator,
chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting,
doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision)
    684     )
    685
--> 686     return _read(filepath_or_buffer, kwds)
    687
    688

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    450
    451     # Create the parser.
--> 452     parser = TextFileReader(fp_or_buf, **kwds)
    453
    454     if chunksize or iterator:

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    934         self.options["has_index_names"] = kwds["has_index_names"]
    935
--> 936         self._make_engine(self.engine)
    937
    938     def close(self):

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _make_engine(self, engine)
    1166     def _make_engine(self, engine="c"):
    1167         if engine == "c":
-> 1168             self._engine = CParserWrapper(self.f, **self.options)
    1169         else:
    1170             if engine == "python":

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in __init__(self, src, **kwds)
    1996         kwds["usecols"] = self.usecols
    1997
-> 1998         self._reader = parsers.TextReader(src, **kwds)
    1999         self.unnamed_cols = self._reader.unnamed_cols
    2000

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] No such file or directory: 'data/safi_clean.csv'

```

```
safi_df2.head()
```

```

-----
NameError                                Traceback (most recent call last)
<ipython-input-6-314725f53798> in <module>
----> 1 safi_df2.head()

NameError: name 'safi_df2' is not defined

```

```
safi_df
```


	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11-17T00:00:00Z	7	40	burntbricks	1	
...
126	126	Ruaca	2017-05-18T00:00:00Z	3	7	burntbricks	1	
127	193	Ruaca	2017-06-04T00:00:00Z	7	10	cement	3	
128	194	Ruaca	2017-06-04T00:00:00Z	4	5	muddaub	1	
129	199	Chirodzo	2017-06-04T00:00:00Z	7	17	burntbricks	2	
130	200	Chirodzo	2017-06-04T00:00:00Z	8	20	burntbricks	2	

131 rows × 14 columns

```
safi_df.to_csv('data/safi_clean.csv',index=False)
```

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-8-8c20409a7eec> in <module>
----> 1 safi_df.to_csv('data/safi_clean.csv', index=False)

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/core/generic.py in to_csv(self, path_or_buf, sep, na_rep,
float_format, columns, header, index, index_label, mode, encoding, compression,
quoting, quotechar, line_terminator, chunksize, date_format, doublequote, escapechar,
decimal, errors)
    3165         decimal=decimal,
    3166     )
-> 3167     formatter.save()
    3168
    3169     if path_or_buf is None:

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/formats/csvs.py in save(self)
    188         encoding=self.encoding,
    189         errors=self.errors,
--> 190         compression=dict(self.compression_args,
method=self.compression),
    191     )
    192     close = True

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/common.py
in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors)
    491     if encoding:
    492         # Encoding
--> 493         f = open(path_or_buf, mode, encoding=encoding, errors=errors,
newline="")
    494     elif is_text:
    495         # No explicit encoding

FileNotFoundError: [Errno 2] No such file or directory: 'data/safi_clean.csv'

```

```

safi_df3 = pd.read_csv('data/safi_clean.csv')
safi_df3.head(3)

```

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-9-a3acle5c7879> in <module>
----> 1 safi_df3 = pd.read_csv('data/safi_clean.csv')
      2 safi_df3.head(3)

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header,
names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine,
converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,
infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator,
chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting,
doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision)
    684     )
    685
--> 686     return _read(filepath_or_buffer, kwds)
    687
    688

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    450
    451     # Create the parser.
--> 452     parser = TextFileReader(fp_or_buf, **kwds)
    453
    454     if chunksize or iterator:

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    934         self.options["has_index_names"] = kwds["has_index_names"]
    935
--> 936         self._make_engine(self.engine)
    937
    938     def close(self):

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in _make_engine(self, engine)
    1166     def _make_engine(self, engine="c"):
    1167         if engine == "c":
-> 1168             self._engine = CParserWrapper(self.f, **self.options)
    1169         else:
    1170             if engine == "python":

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/io/parsers.py in __init__(self, src, **kwds)
    1996         kwds["usecols"] = self.usecols
    1997
-> 1998         self._reader = parsers.TextReader(src, **kwds)
    1999         self.unnamed_cols = self._reader.unnamed_cols
    2000

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] No such file or directory: 'data/safi_clean.csv'

```

```
safi_df['village']
```

```

0      God
1      God
2      God
3      God
4      God
...
126   Ruaca
127   Ruaca
128   Ruaca
129   Chirodzo
130   Chirodzo
Name: village, Length: 131, dtype: object

```

```
safi_df.loc[3]
```

```

key_ID                                4
village                               God
interview_date                        2016-11-17T00:00:00Z
no_membrs                             7
years_liv                             6
respondent_wall_type                  burntbricks
rooms                                 1
memb_assoc                            NaN
affect_conflicts                      NaN
liv_count                             2
items_owned                          bicycle;radio;cow_plough;solar_panel;mobile_phone
no_meals                              2
months_lack_food                      Sept;Oct;Nov;Dec
instanceID                           uuid:148d1105-778a-4755-aa71-281eadd4a973
Name: 3, dtype: object

```

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

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11-17T00:00:00Z	7	40	burntbricks	1	
5	6	God	2016-11-17T00:00:00Z	3	3	muddaub	1	

```
safi_df.loc[:4]
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11-17T00:00:00Z	7	40	burntbricks	1	

```
safi_df.loc[:5]
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
5	6	God	2016-11-17T00:00:00Z	3	3	muddaub	1	
10	11	God	2016-11-21T00:00:00Z	6	20	sunbricks	1	
15	16	God	2016-11-24T00:00:00Z	6	47	muddaub	1	
20	21	God	2016-11-21T00:00:00Z	8	20	burntbricks	1	
25	26	Ruaca	2016-11-21T00:00:00Z	3	20	burntbricks	2	
30	31	Ruaca	2016-11-21T00:00:00Z	3	2	muddaub	1	
35	36	Chirodzo	2016-11-17T00:00:00Z	6	23	sunbricks	1	
40	41	God	2016-11-17T00:00:00Z	7	22	muddaub	1	
45	46	Chirodzo	2016-11-17T00:00:00Z	10	42	burntbricks	2	
50	51	Chirodzo	2016-11-16T00:00:00Z	5	30	muddaub	1	
55	56	Chirodzo	2016-11-16T00:00:00Z	12	23	burntbricks	2	
60	61	Chirodzo	2016-11-16T00:00:00Z	10	14	muddaub	1	
65	66	Chirodzo	2016-11-16T00:00:00Z	10	37	burntbricks	3	
70	71	Ruaca	2016-11-18T00:00:00Z	6	14	burntbricks	1	
75	155	God	2016-11-24T00:00:00Z	4	4	burntbricks	1	

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	m
80	182	God	2016-11-25T00:00:00Z	7	21	muddaub	3	
85	197	God	2016-11-28T00:00:00Z	5	19	burntbricks	2	
90	73	Ruaca	2017-04-26T00:00:00Z	7	9	burntbricks	2	
95	101	God	2017-04-27T00:00:00Z	3	4	muddaub	1	
100	104	Ruaca	2017-04-28T00:00:00Z	14	52	sunbricks	1	
105	113	Ruaca	2017-05-03T00:00:00Z	11	26	burntbricks	3	
110	108	God	2017-05-11T00:00:00Z	15	22	burntbricks	2	
115	150	Ruaca	2017-05-18T00:00:00Z	7	8	muddaub	1	
120	167	Ruaca	2017-06-03T00:00:00Z	8	24	muddaub	1	
125	192	Chirodzo	2017-06-03T00:00:00Z	9	20	burntbricks	1	
130	200	Chirodzo	2017-06-04T00:00:00Z	8	20	burntbricks	2	

```
safi_df['village'].head(2)
```

```
0    God
1    God
Name: village, dtype: object
```

```
safi_df.village.head(2)
```

```
0    God
1    God
Name: village, dtype: object
```

```
safi_df[['village', 'no_membrs', 'years_liv']].head(2)
```

	village	no_membrs	years_liv
0	God	3	4
1	God	7	9

```
columns_of_interest = ['village', 'no_membrs', 'years_liv']
safi_df[columns_of_interest].head(2)
```

	village	no_membrs	years_liv
0	God	3	4
1	God	7	9

Functions are objects

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

```
-----
ImportError                                Traceback (most recent call last)
<ipython-input-19-fac0bcef288c> in <module>
----> 1 syllabus_df_list =
    pd.read_html('https://rhodyprog4ds.github.io/BrownFall20/syllabus/course_map.html')

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-
packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    294         )
    295         warnings.warn(msg, FutureWarning, stacklevel=stacklevel)
--> 296         return func(*args, **kwargs)
    297
    298         return wrapper

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in read_html(io, match, flavor, header, index_col, skiprows, attrs, parse_dates,
thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
    1099     na_values=na_values,
    1100     keep_default_na=keep_default_na,
-> 1101     displayed_only=displayed_only,
    1102 )

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parse(flavor, io, match, attrs, encoding, displayed_only, **kwargs)
    892     retained = None
    893     for flav in flavor:
--> 894         parser = _parser_dispatch(flav)
    895         p = parser(io, compiled_match, attrs, encoding, displayed_only)
    896

/opt/hostedtoolcache/Python/3.7.9/x64/lib/python3.7/site-packages/pandas/io/html.py
in _parser_dispatch(flavor)
    849     else:
    850         if not _HAS_LXML:
--> 851             raise ImportError("lxml not found, please install it")
    852     return _valid_parsers[flavor]
    853

ImportError: lxml not found, please install it
```

```
view_rows = {0: lambda df: print(df.head()),
             1: lambda df: print(df.tail())}
```

```
for df in syllabus_df_list:
    num_row = len(df)
    view_rows[num_row%2](df)
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-21-dc0dd9137832> in <module>
----> 1 for df in syllabus_df_list:
      2     num_row = len(df)
      3     view_rows[num_row%2](df)

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

```
safi_df.describe()
```

	key_ID	no_membrs	years_liv	rooms	liv_count	no_meals
count	131.000000	131.000000	131.000000	131.000000	131.000000	131.000000
mean	85.473282	7.19084	23.053435	1.740458	2.366412	2.603053
std	63.151628	3.17227	16.913041	1.092547	1.082775	0.491143
min	1.000000	2.00000	1.000000	1.000000	1.000000	2.000000
25%	32.500000	5.00000	12.000000	1.000000	1.000000	2.000000
50%	66.000000	7.00000	20.000000	1.000000	2.000000	3.000000
75%	138.000000	9.00000	27.500000	2.000000	3.000000	3.000000
max	202.000000	19.00000	96.000000	8.000000	5.000000	3.000000

```
safi_df.head()
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_
0	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	
2	3	God	2016-11-17T00:00:00Z	10	15	burntbricks	1	
3	4	God	2016-11-17T00:00:00Z	7	6	burntbricks	1	
4	5	God	2016-11-17T00:00:00Z	7	40	burntbricks	1	

```
safi_df.set_index('key_ID',inplace=True)
```

```
safi_df.head(2)
```

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_ass
1	1	God	2016-11-17T00:00:00Z	3	4	muddaub	1	Na
1	1	God	2016-11-17T00:00:00Z	7	9	muddaub	1	yi

```
safi_df.describe()
```

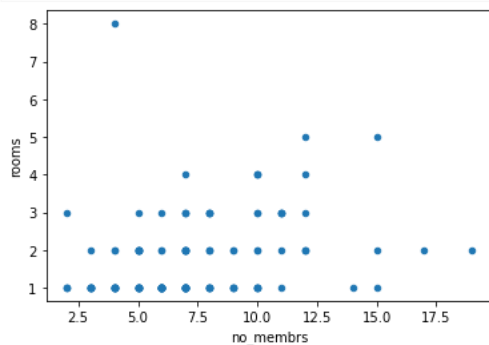

	no_membrs	years_liv	rooms	liv_count	no_meals
count	131.00000	131.000000	131.000000	131.000000	131.000000
mean	7.19084	23.053435	1.740458	2.366412	2.603053
std	3.17227	16.913041	1.092547	1.082775	0.491143
min	2.00000	1.000000	1.000000	1.000000	2.000000
25%	5.00000	12.000000	1.000000	1.000000	2.000000
50%	7.00000	20.000000	1.000000	2.000000	3.000000
75%	9.00000	27.500000	2.000000	3.000000	3.000000
max	19.00000	96.000000	8.000000	5.000000	3.000000

```
safi_df['rooms'].mean()
```

```
1.7404580152671756
```

```
safi_df.plot.scatter('no_membrs', 'rooms')
```

```
<AxesSubplot:xlabel='no_membrs', ylabel='rooms'>
```



After Class Questions

How can we clean data? what other topics will we cover in this class?



How does the syntax on the question from prismia today work?



Where I can go to find a list of all the syntax.



Assignments

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

Assignment TOC:

- [Assignment 1](#) Due September 13
- [Assignment 2](#) Due September 20

Assignment 1: Portfolio Setup, Data Science, and Python

Due: 2020-09-13

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.

```
'''
    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+'
```

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

Due: 2020-09-20

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.

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

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, [Jupytertext](#) 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 column should be the url where you loaded the data from or the source if you downloaded it from a website first The **source_file_name** should be the part of the url after the last /, you should extract this programmatically. Display that summary table as a dataframe and save it as a csv, named **dataset_summary.csv**.

For one dataset (must include 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?

Tip

For a second 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

Tip

Urls are strings. The **string** class in python has a lot of helpful methods for manipulating strings, like [split](#).

Note

If you download the datasets (or find them as .zip and need to) you can use the local path instead of the url, but include a markdown cell with links to where you got your data from.

Tip

You can create a **pandas.DataFrame** using the [constructor](#) and you can build lists (or lists of lists) using the [append](#) method

Portfolio

This section of the site has a set of portfolio prompts and this page has instructions for portfolio submissions.

Starting in week 3 it is recommended that you spend some time each week working on items for your portfolio, that way when it's time to submit you only have a little bit to add before submission.

The portfolio is your only chance to earn Level 3 achievements, however, if you have not earned a level 2 for any of the skills in a given check, you could earn level 2 then instead. The prompts provide a starting point, but remember that to earn achievements, you'll be evaluated by the rubric. You can see the full rubric for all portfolios in the [syllabus](#).

Each submission should include an introduction and a number of 'chapters'. The grade will be based on both that you demonstrate skills through your chapters that are inspired by the prompts and your summary.

The first submission will be graded on the following criteria and due on October 14:

Level 3

keyword	
python	reliable, efficient, pythonic code that consistently adheres to pep8
access	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	merge data that is not automatically aligned
summarize	Compute summary statistics of subsets of data
visualize	generate complex plots with pandas and plotting libraries
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received

On each chapter of your portfolio, you should identify which skills by their keyword, you are applying.

Formatting Tips

Your portfolio is a [jupyter book](#). This means a few things:

- it uses [myst markdown](#)
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

File Naming

It is best practice to name files without spaces.

Configurations

Things like the menus and links at the top are controlled as [settings](#), in `_config.yml`

Links

Markdown syntax for links

```
[text to show](path/or/url)
```

Reflective Prompts

These prompts are more reflective to help demonstrate your understanding of skills. These are more writing than new coding.

Correcting a Prior Assignment

Choose an assignment that you did not achieve the target level for. Write a blog style notebook analysis that corrects what you could have done better, what you learned, and addresses the misconception if applicable.

Data Science Pipeline

Like the day 1 activity, find two different sources that describe the data science pipeline or lifecycle. Write a blog style post that discusses their differences and hypothesizes about why they may be different? Are they for different audiences? Is one domain specific.

Podcast

Watch an episode of a high quality^[1] podcast and write a blog style summary and review of the episode. Highlight what you learned and how it relates to things

Approved Podcasts:

- [Pod of Asclepius, Fall Series: The Philosophy of Data Science](#)

Annotate Class Notes

Annotate class notes by submitting a PR to this repository. This applies after Dr. Brown's notes are uploaded, but not annotated. Use previously annotated notes as an example of what content should be added. In your portfolio, include a 1 paragraph reflection on what you learned by contributing the annotation and a link to your pull request.

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

FAQ

This section will grow as questions are asked and new content is introduced to the site. You can submit questions:

- via e-mail to Dr. Brown (brownsarahm) or Beibhinn (beibhinn)
- via Prismia.chat during class
- by creating an [issue](#)

Note

It's okay if more than one person contributes annotations on the same day. If there's a pull request there already, try to add different, additional insights but if it's not there when you start working and appears before you submit, that's ok.

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 ▾

Resources

This section will compile resources for the course over time into various sections:

- [Data](#) sources of data to use for assignments
- [Programming](#) info on the tools and libraries we're using in class, background info, etc
- [Tips](#) general, semi-related information

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](#)

References on Python

- [Course Text](#)

Data Sources

- [Kaggle](#)
- [Google Dataset Search](#)
- [UCI Data Repository](#)
- [Json Datasets](#)

If you have others please share by creating a pull request or issue on this repo

By Professor Sarah M Brown
© Copyright 2020.