Syllabus

Contents

Welcome to CSC311: Computer Systems and Programming Tools.

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.

Computer Systems and Programming Tools

> **Important**
>
> This syllabus is best viewed as a website see a previous semester's version

About this course

In this course we will study the tools that we use as programmers and use them as a lens to study the computer system itself. We will begin with two fundamental tools: version control and the shell. We will focus on git and bash as popular examples of each. Sometimes understanding the tools requires understanding an aspect of the system, for example git uses cryptographic hashing which requires understanding number systems. Other times the tools helps us see how parts work: the shell is our interface to the operating system. We will survey the computer system across levels of abstraction from applications to hardware. Throughout, we will consider social conventions and practices within programming and the historical development of computer systems.

Learning Outcomes

The goal is for you to learn and the grading is designed to as close as possible actually align to how much you have learned. So, the first thing to keep in mind, always is the course learning outcomes:

By the end of the semester, students will be able to:

1. Apply common design patterns and abstractions to understand new code bases, programming tools, and components of systems.
2. Apply appropriate programming workflows using context-relevant tools that enable adherance to best practices for effective code, developer time efficiency, and collaboration.
3. Differentiate the different classes of tools used in computer science in terms of their features, roles, and how they interact and justify positions and preferences among popular tools
4. Identify how information flows across levels of abstraction.
5. Discuss implications of design choices across levels of abstraction
6. Describe the social context in which essential components of computing systems were developed and explain the impact of that context on the systems.
7. Differentiate between social conventions and technical requirements in programming contexts.

About your instructor

- Name: Dr. Sarah M Brown
- email: brownsarahm@uri.edu
- github: brownsarahm
- Office hours: listed on communication page and GitHub Org member view

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

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

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

Land Acknowledgement

> **Important**
>
> The University of Rhode Island land acknowledgment is a statement written by members of the University community in close partnership with members of the Narragansett Tribe. For more information see the university land acknowledgement page

The University of Rhode Island occupies the traditional stomping ground of the Narragansett Nation and the Niantic People. We honor and respect the enduring and continuing relationship between the Indigenous people and this land by teaching and learning more about their history and present-day communities, and by becoming stewards of the land we, too, inhabit.

Format and Schedule

Structure

This class is cerntered on active learning. Over the course of the semester, we will cover topics through multiple lenses so that you get multiple chances to build skill with the tools that we are learning and see how the important concepts repeat in different ways throughout

There will prescribed preparation work for each class session that either helps you preview tricky topics in advance or refresh your memory of prerequisite material so that you are ready to learn the new material.

In class we will focus on answering a new question together each session through a combination of inquiry-driven participatory live coding and discussions. You will be expected to follow along with the activity and answer questions to check understanding.

After each class there will 2 sets of activities based on that material, you can choose which level to work at. The review level activities require you to review what was done in class and ensure that you understand it. The practice level tasks give you a chance to get more practice and apply your understanding in new ways beyond what we do in class.

Topics

This is the planned schedule, but is subject to change in order to adapt to how things go in class or additional questions that come up.

Each class we will answer a motivating question in a way that integrates important practical skills, conceptual knowledge, and social context of computer systems and programming tools.

We will focus on *learning* the tools to start, treating them as an example of how computer systems are designed and then *using* the tools to study the system in the second half of the semester.

date	Question	Keyword	Activity
2023-09-07	Welcome, Introduction, and Setup	intro	create kwl repo in github, navigate github.com...
2023-09-12	Course Logistics and Learning	logistics	set up to work offline together, create a folder
2023-09-14	Bash intro & git offline	terminal start	navigate files and clone a repo locally
2023-09-19	How can I work with branches offline?	gitoffline	clone a repo and make a branch locally
2023-09-21	When do I get an advantage from git and bash?	why terminal	work with bash and recover from a mistake with...
2023-09-26	What *is* a commit?	merge conflicts	examine commit objects, introduce plumbing com...
2023-09-28	How do programmers communicate about code?	documentation	make a jupyterbook
2023-10-03	What *is* git?	git structure	examine git from multiple definitions and insp...
2023-10-05	Why are these tools like this?	unix philosophy	discussion with minor code examples
2023-10-12	How does git make a commit?	git internals	create a commit using plumbing commands
2023-10-17	How can can I release and share my code?	git references	make a tag and release
2023-10-19	What is a commit number?	numbers	discussion and use hashing algorithm
2023-10-24	How can I automate things with bash?	bash scripting	build a bash script that calculates a grade
2023-10-26	How can I work on a remote server?	server	log into a remote server and work with large f...
2023-10-31	What is an IDE?	IDE	discussions and sharing IDE tips
2023-11-02	How do I choose a Programming Language for a p...	programming languages	discussion or independent research
2023-11-07	How can I authenitcate more securely from a te...	server use	configure and use ssh keys on a hpc
2023-11-09	What Happens when we build code?	building	build code in C and examine intermediate outputs
2023-11-14	What happens when we run code?	hardwar	use a hardware simulator to see step by step o...
2023-11-16	How does a computer represent non integer quan...	floats	work with float representation through fractio...
2023-11-21	How can we use logical operations?	bitwise operation	derive addition from basic logic operations
2023-11-28	What *is* a computer?	architecture	discussion

date	Question	Keyword	Activity
2023-11-30	How does timing work in a computer?	timing	write a threaded program and fix a race condition
2023-12-05	How do different types of storage work together?	memory	large data that has to be read in batches
2023-12-07	How does this all work together	review	review quiz, integration/reflection questions
2023-12-12	How did this semester go?	feedback	discussion

Lab Schedule

Lab provides extra support and accountability. Some lab activities will directly be time to work on tasks from review or practice badges. Some lab activities will help you review class material. Some lab activities will include self-assessments and reflection on your course progress to ensure that you are not falling behind.

Every week at the end of lab you will “check out” with a TA or instructor, reviewing your work and showing that you completed any setup or install steps that are needed for the upcoming classes. The checkout process requires that you have a brief conversation with the instructor or a TA to show that you have have completed steps and are prepared for upcoming classes. For example, that you know what the terminal to use is for the upcoming material on your specific computer since that can vary from student to student. The checkout is evaluated only on completion, any components that are important to get correct will be assessed in a review or practice badge.

▶ Show code cell source

date	topic	activity
2023-09-08	GitHub Basics	syllabus quiz, setup
2023-09-15	working at the terminal	organization, setup kwl locally, manage issues
2023-09-22	offline branches	plan for success, clean a messy repo
2023-09-29	tool familiarity	work on badges, self progress report
2023-10-06	unix philosophy	design a command line tool that would enable a...
2023-10-13	git plumbing	git plumbing experiment
2023-10-20	git plumbing	grade calculation script, self reflection
2023-10-27	scripting	releases and packaging
2023-11-03	remote, hpc	server work, batch scripts
2023-11-10	Compiling	C compiling experiments
2023-11-17	Machine representation	bits and floats and number libraries
2023-12-01	hardware	self-reflection, work, project consultations
2023-12-08	os	hardware simulation

Tools and Resources

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

All of the tools and resources below are either:

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

BrightSpace

On BrightSpace, you will find links to essential resources for the first day of class, including this site. Any links that are for private discussion among those enrolled in the course will be available only from Brightspace, not this site.

Prismia chat

Our class link for Prismia chat is available on Brightspace. Once you've joined once, you can use the link above or type the url: `prismia.chat``. 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.

> **Important**
>
> Prismia is **only** for use during class, we do not read messages there outside of class time

You can get a transcript from class from Prismia.chat using the menu in the top right.

Course Website

The course website will have content including the class policies, scheduling, class notes, assignment information, and additional resources.

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

> **Important**
>
> There is no text book for the course. Reading the notes posted on the website and any freely available external readings linked in prepare for class is required. Supplemental texts and videos will be linked in the notes to provide greater details or alternative explanations.

GitHub

You will need a GitHub Account. If you do not already have one, please create one before the first class session. If you have one, but have not used it recently, you may need to update your password and login credentials as the Authentication rules change

> **Not**
>
> Seeir
> loggir
> being

often and now require 2 factor.

Programming Environment

In this course, we will use several programming environments. In order to participate in class and 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, however if you have alternative environments they will mostly work. We will add tools throughout the semester, but the following will be enough to get started.

> **Warning**
>
> This is not technically a *programming* class, so you will not need to know how to write code from scratch in specific languages, but we will rely on programming environments to apply concepts.

Requirements:

- Python with scientific computing packages (numpy, scipy, jupyter, pandas, seaborn, sklearn)
- a C compiler
- Git
- A `bash` shell
- A web browser compatible with Jupyter Notebooks
- nano text editor (comes with GitBash and default on MacOS)
- one IDE with git support (default or via extension)
- the GitHub CLI or other authentication for GitHub

Recommendation

Windows (work with it) | **Windows (get linux)** | **MacOS** | **Linux** | **Chrome OS**

- Install python via Anaconda video install
- Git and Bash with GitBash (video instructions).

Zoom

> **Note**
>
> (backup & office hours only)

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

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 office hours and any online class sessions if needed best if you download the Zoom client on your computer. Please log in and configure your account. Please add a photo (can be yourself or something you like) 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.

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

Grading

This section of the syllabus describes the principles and mechanics of the grading for the course. The course is designed around your learning so the grading is based on you demonstrating how much you have learned, but not based on how quickly you learn it.

Additionally, since we will be studying programming tools, we will use them to administer the course. To give you a chance to get used to the tools there will be a penalty free zone for the first few weeks.

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 designed to reflect how deeply you learn the material, even if it takes you multiple attempts to truly understand a topic. The topics in this course are all topics that will come back in later courses in the Computer Science major, so it is important that you understand each of them *correctly* so that it helps in the next course.

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 material. You will be required to demonstrate understanding of the connections between ides from different parts of the course.

- Earning a C in this class means you have a general understanding; you will know what all the terms mean; you could follow along in a meeting where others were discussing systems concepts and use core tools for common tasks. You know where to start when looking things up.
- Earning a B means that you can apply the course concepts in other programming environments; you can solve basic common errors without looking much up.
- Earning an A means that you can use knowledge from this course to debug tricky scenarios; you know where to start and can form good hypotheses about why uncommon errors have occurred; you can confidently figure out new complex systems.

The course is designed for you to *succeed* at a level of your choice. As you accumulate knowledge, the grading in this course is designed to be cumulative instead of based on deducting points and averaging. No matter what level of work you choose to engage in, you will be expected to revise work until it is correct. The material in this course will all come back in other 300 and 400 level CSC courses, so it is essential that you do not leave this course with misconceptions, as they will make it harder for you to learn related material later.

Penalty-free Zone

Since learning developer tools is a core learning outcome of the course, we will also use them for all aspects of administering the course. This will help you learn these tools really well and create accountability for getting enough practice with core operations, but it also creates a high stakes situation: even submitting your work requires you understanding the tools. This would not be very fair at the beginning of the semester.

For the first three weeks we will have a low stakes penalty-free zone where we will provide extra help and reminders for how to get feedback on your work. In this period, deadlines are more flexible as well. If work is submitted incorrectly, we will still see it

because we will manually go look for all activities. After this zone, we will assume you *chose* to skip something if we do not see it.

> **Important**
>
> If there are terms in the rest of this section that do not make sense while we are in the penalty-free zone, do not panic. This zone exists to help you get familiar with the terms needed to succeed in the course.

During the third week, you will create a course plan where you establish your goals for the course and I make sure that you all understand the requirements to complete your goals.

Learning Badges

Your grade will be based on you choosing to work with the material at different levels and participating in the class community in different ways. Each of these represents different types of badges that you can earn as you accumulate evidence of your learning and engagment.

- experience: guided in class acitivies
- review: just the basics
- practice: a little bit more indepdendent
- explore: posing your own directions of inquiry
- build: in depth- application of course topics

Review and practice badges are dated and you can only earn one of the two for each date. All of these badges will be tracked through PRs in your kwl repo. Each PR must have a title that includes the badge type and associated date. We will use scripts over these to track your progress.

To earn a D you must complete:

- 22 experience badges
- 13 lab check outs

To earn a C you must complete:

- 22 experience badges
- 13 lab check outs
- 18 review badges

To earn a B you must complete:

- 22 experience badges
- 13 lab check outs
- your choice:
  - 18 practice badges
  - 12 review + 12 practice

For an A you must complete:

- 22 experience badges
- 13 lab check outs

- your choice:
  - 18 practice badges + 6 explore badges
  - 18 review badges + 3 build badges
  - 6 review badges + 12 practice badges + 4 explore badges + 1 build badges
  - 12 review badges + 6 practice badges+ 2 explore badges + 2 build badges

You can also mix and match to get +/-. For example (all examples below assume 22+ experience badges aand 13 lab checkouts)

- A-: 18 practice + 4 explore
- B+: 6 review + 12 practice + 4 explore
- B-: 6 review + 12 practice
- B+: 24 practice
- C+: 12 review + 6 practice

Experience Badges

In class

You earn an experience badge in class by:

- preparing for class
- following along with the activity (creating files, using git, etc)
- responding to 80% of inclass questions (even incorrect answers or `\:idk:` count here)
- reflecting on what you learned and asking a question at the end of class in your experience reflection

Makeup

You can make up an experience badge by:

- preparing for class
- reading the posted notes
- completing the activity from the notes
- completeing an “experience report”
- attaching evidence as indiated in notes OR attending office hours to show the evidence

An experience report is evidence you have completed the activity and reflection questions. The exact form will vary per class, if you are unsure, reach out ASAP to get instructions. These are evaluated only for completeness/ good faith effort. Revisions will generally not be required, but clarification and additional activity steps may be advised if your evidence suggests you may have missed a step.

Review and Practice Badges

The tasks for these badges will be defined at the bottom of the notes for each class session, aggregated by type on the course website, and issues will be created for each badge in your KWL repo where you track your work.

You can earn review and practice badges by:

- completing the tasks
- submitting files to your KWL on a badge-specific branch
- creating a PR, linking the issue, and requesting a review
- revising the PR until it is approved
- merging the PR after it is approved

Where do you find assignments?

At the end of notes and on the separate pages in the activities section on the left hand side

You should create one PR per badge

The key difference between review and practice is the depth of the activity. Work submitted for review and practice badges will be assessed for correctness and completeness. Revisions will be common for these activities, because understanding correctly, without misconceptions, is important.

Important

Revisions are to help you improve your work **and** to get used to the process of making revisions. Even excellent work can be improved. The **process** of making revisions and taking good work to excellent or excellent to exceptional is a useful learning outcome. It will help you later to be really good at working through PR revisions; we will use the same process as code reviews in industry, even though most of it will not be code alone.

Explore Badges

Explore badges require you to pose a question of your own that extends the topic. For inspiration, see the practice tasks and the questions after class.

Details and more ideas are on the explore page of the course website.

For these, ideas will almost always be approved, the proposal is to make sure you have the right scope (not too big or too small). If your proposal is too big, you will get advice to either cut it back or transform it to a build badge. If it is too small, you will get advice on how to flesh it out a little bit. Work submitted for explore badges will be assessed for depth beyond practice badges and correctness. Revisions will be more common on the first few as you get used to them, but typically decraese as you learn what to expect.

You should create one PR per badge

Build Badges

Build badges are for when you have an idea of something you want to do. There are also some ideas on the build page.

For builds, since they’re bigger, you will propose intermediate milestones. Advice for improving your work will be provided at the milestones and revisions of the complete build are uncommon. If you do not submit work for intermediate review, you may need to revise the complete build. The build proposal will assessed for relevance to the course and depth. The work will be assessed for completeness in comparison to the propsal and correctness. The summary report will be assessed only for completeness, revisions will only be requested for skipped or incomplete sections.

Community Badges

Community badges are awarded for extra community participation. Both programming and learning are most effective with good collaboration. Since being a good member of our class community helps you learn (and helps others learn better), some collaboration is required in other badges. Some dimensions of community participation can only be done once, for example fixing a typo on the course website, so while it's valuable, all students cannot contribute to the course community in the same way. To reward these unique contributions, you can earn a community badge.

You can see some ideas as they arise by issues labeled `community` on the course website repo.

Community badges can replace missed experience, review, and practice badges, upgrade a review to a practice badge, or they can be used as an alternate way to earn a + modifier on a D,C,or B (URI doesn't award A+s, sorry). Community badges are smaller, so they are not 1:1 replacements for other badges. You can earn a maximum of 14 community badges, generally one per week. Extra helpful contributions may be awarded 2 community badges, but that does not increase your limit. When you earn them, you can plan how you will use it, but they will only be officially applied to your grade at the end of the semester. They will automatically be applied in the way that gives you the maximum benefit.

Community Badge values:

- 3 community = 1 experience badge
- 4 community = 1 review
- 7 community = 1 practice.
- 3 community badges + 1 review = 1 practice.
- 10 community = add a `+` to a D,C, or B, **note that this is more efficient.**

You can earn community badges by:

- fixing small issues on the course website (during penalty free zone only)
- contributing extra terms or reviews to your team repo
- sharing articles and discussing them in the course discussions
- contributing annotated resources the course website

You will maintain a list of your contributions in your KWL repo in the `community_contributions.md` file. Every individual change to this file (representing one contribution) should be commited to a new branch and then submitted as a PR, with a review requested from @brownsarahm.

> **Note**
>
> Some participation in your group repo and a small number of discussions will be required for experience, review, and practice badges. This means that not every single contribution or peer review to your team repo will earn a community badge.

Example(nonexhaustive) uses:

- 22 experience + 17 review + 11 community = C (replace 2 experience, 1 review)
- 24 experience + 17 review + 5 community = C (replace 1 review)
- 24 experience + 18 review + 10 community = C+ (modifier)
- 24 experience + 18 practice + 10 community = B+ (modifier)
- 23 experience + 18 practice + 13 community = B+ (modifier, replace 1 experience)

- 24 experience + 16 practice + 2 review + 10 community = B (upgrade 2 review)
- 24 experience + 10 review + 10 community + 6 practice + 3 explore + 2 build = A (replace 2 review)
- 24 experience + 14 review + 10 community + 4 practice + 3 explore + 2 build = A (upgrade 2 review to practice)
- 24 experience + 12 review + 14 community + 4 practice + 3 build =A (replace 2 practice)

These show that community badges can save you work at the end of the semester by reducing the number of practice badges or simplifying badges

🎁 Free corrections

All work must be correct and complete to earn credit. In general, this means that when your work is not correct, we will give you guiding questions and advice so that you can revise the work to be correct. Most of the time asking you questions is the best way to help you learn, but sometimes, especially for small things, showing you a correct example makes the most sense.

Additionally, on rare occasions, a student can submit work that is incorrect or will have down-the-line consquences but does not demonstrate a misunderstanding. For example, in an experience badge, putting text below the `#` line instead of replacing the hint within the `< >`. Later, we will do things within the kwl repo that will rely on the title line being filled in, but it's not a big revision where the student needs to rethink about what they submitted.

In these special occasions, good effort that is not technically correct may be rewarded with a 🎁. In this case, the instructor or TA will give a suggestion, with the 🎁 emoji in the comment and leave a review as "comment" instead of "changes requested" or "approved". If the student commits the suggestion to acknowledge that they read it, the instructor will then leave an approving review. Free corrections are only available when revisions are otherwise eligible. This means that they cannot extend a deadline and they are not available on the final grading that occurs after our scheduled "exam time".

> **Important**
>
> These free corrections are used at the instructional team's discretion and are not guaranteed.
>
> This means that, for example, the same mistake the first time, might get a 🎁, a second will probably be a hint, and a third or fourth time might be a regular revision where we ask you to go review prior assignments to figure out what you need to fix with a broad hint instead of the specific suggestion.

Deadlines

Experience badges

Prepare for class tasks must be done before class so that you are prepared. Missing a prepare task could require you to do an experience report to make up what you were not able to do in class.

If you miss class, the experience report should be at least attempted/drafted (though you may not get feedback/confirmation) before the next class that you attend. This is strict, not as punishment, but to ensure that you are able to participate in the next class that you attend. Skipping the experience report for a missed class, may result in needing to do an experience report for the next class you attend to make up what you were not able to complete due to the missing class activities.

If you miss multiple classes, create a catch-up plan to get back on track by contacting Dr. Brown.

Review and Practice Badges

These badges have 5 stages:

- posted: tasks are on the course website and issue created
- started: one task is attempted and a draft PR is open
- completed: all tasks are attempted PR is ready for review, and a review is requested
- earned: PR is approved (by instructor or a TA) and work is merged

These badges badges must be *started* within one week of when the are posted (2pm) and *completed* within two weeks. A task is attempted when you have answered the questions or submitted evidence of doing an activity or asked a sincere clarifying question.

If a badge is not started within one week it will become expired and ineligble to be earned. You may request extensions to complete a badge by updating the PR message, these will typically be granted. Extensions for starting badges will only be granted in exceptional circumstances.

Expired badges will receive a comment and be closed.

Once you have a good-faith attempt at a complete badge, you have until the end of the semester to finish the revisions in order to *earn* the badge.

Explore Badges

Explore badges have 5 stages:

- proposed: issue created
- in progress: issue is labeled “proceed” by the instructor
- complete: work is complete, PR created, review requested
- revision: “request changes” review was given
- earned: PR approved

Explore badges are feedback-limited. You will not get feedback on subsequent explore badge proposals until you earn the first one. Once you have one earned, then you can have up to two in progress and two in revision at any given time.

Build Badges

You may earn at most one build badge per month, with final grading in December. To earn three build badges, you must earn the first one by the end of October.

Ungrading Option

At the end of the semester, you have the option of submitting a final reflection that states what grade you think you deserve, and justifies it by summarizing what you have learned and providing evidence of that. Instructions for this option will be provided as we approach the end of the semester. The policy of no submitted content that was not generated by you still applies. If you take this option, you may be required to also take an oral exam by appointment to supplement the evidence provided in your reflection.

This option exists in recognition of the fact that grading schemes are not perfect and I am truly committed to your learning. If you think that the grading scheme described on this page is working out to you earning a different grade than you deserve and you can

support that with strong evidence that you have learned, you can have the grade you deserve.

Academic Honesty

All of your work must reflect your own thinking and understanding. The work that you submit must all be your own work or content that was provided to you in class, it cannot include text that was generated by an AI or plagiarized in any other way. You may use auto-complete in all tools and GitHub co-pilot for any code that is required for this course because the code is necessary to demonstrate examples, but language syntax is not the core learning outcome.

If you are found to submit prismia responses that do not reflect your own thinking or that of discussion with peers as directed, the experience badge for that class session will be ineligible.

If work is suspected, you will be allowed to take an oral exam in lab time to contest and prove that your work reflects your own understanding.

The first time you will be allowed to appeal through an oral exam. If your appeal is successful, your counter resets. If you are found to have violated the policy then the badge in question will be ineligible and your maximum number of badges possible to be earned will be limited (you cannot treat the plagiarized badge as skipped).

If you are found to submit work that is not your own for a review or prepare badge, the review and prepare badges for that date will be ineligible and the penalty free zone terms will no longer apply to the first six badges.

If you are found to submit work that is not your own for an explore or build badge, that badge will not be awarded and your maximum badges at the level possible will drop to 2/3 of the maximum possible.

Support

Academic Enhancement Center

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

- **STEM Tutoring** helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses. The STEM Tutoring program offers free online and limited in-person peer-tutoring this fall. Undergraduates in introductory STEM courses have a variety of small group times to choose from and can select occasional or weekly appointments. Appointments and locations will be visible in the TutorTrac system on September 14th, FIXME. 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 FIXME, all Academic Skills and Strategies programming are offered both online and in-person. UCS160: Success in Higher Education is a one-credit course on developing a more effective approach to studying. Academic Consultations are 30-minute, 1 to 1 appointments that students can schedule on Starfish with Dr. David Hayes to address individual academic issues. Study Your Way to Success is a self-guided web portal connecting students to tips and strategies on studying and time management

related topics. For more information on these programs, visit the Academic Skills Page or contact Dr. Hayes directly at davidhayes@uri.edu.

- The **Undergraduate Writing Center** provides free writing support to students in any class, at any stage of the writing process: from understanding an assignment and brainstorming ideas, to developing, organizing, and revising a draft. Fall 2020 services are offered through two online options: 1) real-time synchronous appointments with a peer consultant (25- and 50-minute slots, available Sunday - Friday), and 2) written asynchronous consultations with a 24-hour turn-around response time (available Monday - Friday). Synchronous appointments are video-based, with audio, chat, document-sharing, and live captioning capabilities, to meet a range of accessibility needs. View the synchronous and asynchronous schedules and book online, visit uri.mywconline.com.

General URI Policies

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, Access, and Inclusion Services for Students Statement

Your access in this course is important. Please send me your Disability, Access, and Inclusion (DAI) 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 DAI, please contact them to engage in a confidential conversation about the process for requesting reasonable accommodations in the classroom. DAI can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dai@etal.uri.edu. We are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

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

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

- Submitting the same paper for more than one course without prior approval from the instructors

Viral Illness Precautions

The University is committed to delivering its educational mission while protecting the health and safety of our community. Students who are experiencing symptoms of viral illness should NOT go to class/work. Those who test positive for COVID-19 should follow the isolation guidelines from the Rhode Island Department of Health and CDC.

If you miss class, you do not need to notify me in advance. You can follow the makeup procedures

Excused Absences

Absences due to serious illness or traumatic loss, religious observances, or participation in a university sanctioned event are considered excused absences. For this, contact Dr. Brown when you are ready to get caught up and she will help you make a plan for the best order to complete missed work so that you are able to participate in subsequent activities. Extensions on badges will be provided if needed.

Mental Health and Wellness:

We understand that college comes with challenges and stress associated with your courses, job/family responsibilities and personal life. URI offers students a range of services to support your mental health and wellbeing, including the URI Counseling Center, MySSP (Student Support Program) App, the Wellness Resource Center, and Well-being Coaching.

Office Hours & Communication

Announcements

Announcements will be made via GitHub Release. You can view them online in the releases page or you can get notifications by watching the repository, choosing “Releases” under custom see GitHub docs for instructions with screenshots. You can choose GitHub only or e-mail notificaiton from the notification settings page

Help Hours

Day	Time	Location	Host
Monday	11-1	Zoom	TA
Monday	4-5	Zoom	Dr. Brown
Tuesday	11:15-12:30	Tyler 140	TA
Wednesday	1-2	Zoom	TA
Thursday	11:15-12:30	Tyler 140	TA
Friday	10-11	Zoom	TA
Friday	4-5	Tyler 134	Dr. Brown

Online office hours locations are linked on the GitHub Organization Page

> **Important**
>
> You can only see them if you are a “member” to join, use the “Whole Class Discussion” link in prismia from the first class.

Tips

For assignment help

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

Using issues

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

> **...**
>
> You can submit a pull request for the typo above, but be sure to check the pull request tab of the repo before submitting to see if it has already been submitted.

For E-mail

- use e-mail for general inquiries or notifications
- Please include `[CSC392]` 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 include that in subject to ensure that I see it.

> **Should you e-mail your work?**
>
> No, request a pull request review or make an issue if you are stuck