# Syllabus

## Contents

Syllabus

- Computer Systems and Programming Tools
- Schedule
- Tools and Resources
- Grading
- Support
- General Policies
- Office Hours & Communication

Welcome to CSC392: 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.

This is a live document that will change over time, but a pdf copy is available for direct download or to view on GitHub. If you download it, it will be outdated.

## Computer Systems and Programming Tools

### 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.

### About this syllabus

This syllabus is a *living* document. 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 commit history page.

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

Skip to main content

**Should you download the syllabus and rely on your offline copy?**

No, because the syallabus changes

## About your instructor

Name: Dr. Sarah M Brown Office hours: listed on communication page

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

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

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

## 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.

# Schedule

## Overview

The following is a tentative outline of topics in an order, these things will be filled into the concrete schedule above as we go. These are, in most cases bigger questions than we can tackle in one class, but will give the general idea of how the class will go.

### How does this class work?

*one week*

We'll spend the first two classes introducing some basics of GitHub and setting expectations for how the course will work. This

Skip to main content

in computer

science works and getting started with the programming tools.

## What tools do Computer Scientists use?

Next we'll focus in on tools we use as computer scientists to do our work. We will use this as a way to motivate how different aspects of a computer work in greater detail. While studying the tools and how they work, we will get to see how some common abstractions are re-used throughout the fields and it gives a window and good motivation to begin considering how the computer actually works.

Topics:

- bash
- linux
- git
- i/o
- ssh and ssh keys
- number systems
- file systems

## What Happens When I run code?

Finally, we'll go in really deep on the compilation and running of code. In this part, we will work from the compilation through to assembly down to hardware and then into machine representation of data.

Topics:

- software system and Abstraction
- programming languages
- cache and memory
- compiliation
- linking
- basic hardware components

# Recommended workload distribution

> **Note**
>
> General badge deadlines are on the detailed badge procedures page.

To plan your time, I recommend expecting the following:

- 30 minutes, twice per week for prepare work (typically not this much).
- 1.5(review)-3(practice) hours, twice per week for the dated badges (including revisions).

Skip to main content

For each explore:

- 30 min for proposal
- 7 hours for the project

For each build:

- 1.5 hour for the proposal (including revisions)
- 22 hours for the project
- 30 min for the final reflection

This is a four credit course, meaning we have approximately 4 hours of class + lab time per week($75 \times 2 + 105 = 255$ minutes or 4.25 hours). By the accredidation standards, students should spend a minimum of 2 hours per credit of work outside of class over 14 weeks. For a 4 credit class, then, the expected minimum number of hours of work outside of class you should be spending is 112 hours(2 * 4 * 14). With these calculations, given that there are 26 class sessions and only 18 review or practice are required, it is possible to earn an A with approximately 112 hours of work outside of class and lab time.

## Tentative Timeline

> **Warning**
>
> This section is not yet updated for spring 2024.
>
> This is a rough example.

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.

```
import pandas as pd
pd.read_csv('schedule.csv',index_col='date').sort_index()
```

Skip to main content

| date | question | keyword | conceptual | practical | social | activity |
|---|---|---|---|---|---|---|
| **2023-09-07** | Welcome, Introduction, and Setup | intro | what is a system, why study tools | GitHub basics | class intros | create kwl repo in github, navigate github.com... |
| **2023-09-12** | Course Logistics and Learning | logistics | github flow with issues | syllabus | working together and building common vocab | set up to work offline together, create a folder |
| **2023-09-14** | Bash intro & git offline | terminal start | git structure, paths and file system | bash path navigation, git terminal authentication | why developers work differently than casual users | navigate files and clone a repo locally |
| **2023-09-19** | How can I work with branches offline? | gitoffline | git branches | github flow offline, resolving merge conflicts | commuication is important, git can help fix mi... | clone a repo and make a branch locally |
| **2023-09-21** | When do I get an advantage from git and bash? | why terminal | computing mental model, paths and file structure | bash navigation, tab completion | collaboration requires shared language, shared... | work with bash and recover from a mistake with... |
| **2023-09-26** | What *is* a commit? | merge conflicts | versions, git vlaues | merge conflicts in github, merge conflicts wit... | human and machine readable, commit messages ar... | examine commit objects, introduce plumbing com... |
| **2023-09-28** | How do programmers communicate about code? | documentation | build, automation, modularity, pattern matching, | generate documentation with jupyterbook, gitig... | main vs master, documentation community | make a jupyterbook |
| **2023-10-03** | What *is* git? | git structure | what is a file system, how does git keep track... | find in bash, seeing git config, plumbing/porc... | git workflows are conventions, git can be used... | examine git from multiple definitions and insp... |
| **2023-10-05** | Why are these tools like this? | unix philosophy | unix philosophy, debugging strategies | decision making for branches | social advantages of shared mental model, diff... | discussion with minor code examples |
| **2023-10-12** | How does git make a commit? | git internals | pointers, design and abstraction, intermediate... | inspecting git objects, when hashes are unique... | conventions vs requirements | create a commit using plumbing commands |
| **2023-10-17** | What is a commit number? | numbers | hashes, number systems | git commit numbers, manual hashing with git | number systems are derived in culture | discussion and use hashing algorithm |
| **2023-10-19** | How can can I release and share my code? | git references | pointers, git branches and tags | git branches, advanced fixing, semver and conv... | advantages of data that is both human and mach... | make a tag and release |
| **2023-10-24** | How can I automate things with bash? | bash scripting | bash is a programming language, official docs,... | script files, man pages, bash variables, bash ... | using automation to make collaboration easier | build a bash script that calculates a grade |

Skip to main content

log into a remote erver and work with

| date | question | keyword | conceptual | practical | social | activity |
|---|---|---|---|---|---|---|
| 2023-10-31 | What is an IDE? | IDE | IDE parts | compare and contrast IDEs | collaboraiton features, developer communities | discussions and sharing IDE tips |
| 2023-11-02 | How do I choose a Programming Language for a p... | programming languages | types of PLs, what is PL studying | choosing a language for a project | usability depends on prior experience | discussion or independent research |
| 2023-11-07 | How can I authenitcate more securely from a te... | server use | ssh keys, hpc system strucutre | ssh keys, interactive, slurm | social aspects of passwords and security | configure and use ssh keys on a hpc |
| 2023-11-09 | What Happens when we build code? | building | building C code | ssh keys, gcc compiler | file extensions are for people, when vocabular... | build code in C and examine intermediate outputs |
| 2023-11-14 | What happens when we run code? | hardwar | von neuman architecture | reading a basic assembly language | historical context of computer architecures | use a hardware simulator to see step by step o... |
| 2023-11-16 | How does a computer represent non integer quan... | floats | float representation | floats do not equal themselves | social processes around standard developents, ... | work with float representation through fractio... |
| 2023-11-21 | How can we use logical operations? | bitwise operation | what is a bit, what is a register, how to brea... | how an ALU works | tech interviews look for obscure details somet... | derive addition from basic logic operations |
| 2023-11-28 | What *is* a computer? | architecture | physical gates, history | interpretting specs | social context influences technology | discussion |
| 2023-11-30 | How does timing work in a computer? | timing | timing, control unit, threading | threaded program with a race condition | different times matter in different cases | write a threaded program and fix a race condition |
| 2023-12-05 | How do different types of storage work together? | memory | different type of memory, different abstractions | working with large data | privacy/respect for data | large data that has to be read in batches |
| 2023-12-07 | How does this all work together | review | all | end of semester logistics | group work final | review quiz, integration/reflection questions |
| 2023-12-12 | How did this semester go? | feedback | all | grading | how to learn better together | discussion |

# Tentative Lab schedule

```
pd.read_csv('labschedule.csv',index_col='date').sort_index()
```

Skip to main content

| 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 other resource, this site and others. Any links that are for private discussion among those enrolled in the course will be available only from Brightspace.

## 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.

Skip to main content

> **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.

## GitHub

You will need a GitHub Account. If you do not already have one, please create one by the first day of class. If you have one, but have not used it recently, you may need to update your password and login credentials as the Authentication rules changed in Summer 2021.

You will also need the gh CLI. It will help with authentication and allow you to work with other parts of GitHub besides the core git operations.

> **Important**
>
> You need to install this on Mac

## 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. 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

Skip to main content

- 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 on all OSs

### Recommendation

**Windows- option A** | **Windows - option B** | **MacOS** | **Linux** | **Chrome OS**

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

## Zoom

(backup only & 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 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 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.

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 grade free zone for the first few weeks.

## 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

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.

These are what I will be looking for evidence of to say that you met those or not.

## 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 an A means that you can use knowledge from this course to debug tricky scenarios; you can know where to start and can form good hypotheses about why uncommon errors have occurred; you can confidently figure out new complex systems.
- 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 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.

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.

**If you made an error in an assignment what do you need to do?**

Read the suggestions and revise the work until it is correct.

Skip to main content

# 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 submitted.

**What happens if you merged a PR without feedback?**

During the Penalty-Free zone, we will help you figure that out and fix it so you get credit for it. After that, you have to fix it on your own (or in office hours) in order to get credit.

**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.

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.

**What happens if you're confused by the grading scheme right now?**

Nothing to worry about, we will review it again in week three after you get a chance to build the right habits and learn vocabulary. There will also be a lab activity that helps us to be sure that you understand it at that time.

# 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 form of engagment is tracked with a particular type of badge that you can earn as you accumulate evidence of your learning and engagment.

- experience: guided in class activities, with reflection
- lab: accountability to basics through 1:1 conversation with a member of the instructional team
- review: just the basics
- practice: a little bit more indepdendent
- explore: posing your own directions of inquiry
- build: in depth application of course topics

Skip to main content

To earn a D, complete:

- 22 experience badges
- 13 lab check outs

To earn a C, complete:

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

To earn a B, complete:

- 22 experience badges
- 13 lab check outs
- one of the following:
  - 18 practice badges
  - 12 review + 12 practice

For an A you must complete:

- 22 experience badges
- 13 lab check outs
- one of the following:
  - 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

> **Warning**
>
> These counts assume that the semester goes as planned and that there are 26 available badges of each base type (experience, review, practice). If the number of available badges decreases by more than 2 for any reason (eg snowdays, instructor illness, etc) the threshold for experience badges will be decreased.

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

Skip to main content

**Important**

There will be 20 review and practice badges available after the penalty free zone. This means that missing the review and practice badges in the penalty free zone cannot hurt you. However, it does not mean it is a good idea to not attempt them, not attempting them at all will make future badges harder, because reviewing early ideas are important for later ideas.

You cannot earn both practice and review badges for the same class session, but most practice badge requirements will include the review requirements plus some extra steps.

In the second half of the semester, there will be special *integrative* badge opportunities that have multipliers attached to them. These badges will count for more than one. For example an integrative 2x review badge counts as two review badges. These badges will be more complex than regular badges and therefore count more.

**Can you do any combination of badges?**

No, you cannot earn practice and review for the same date.

## 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, `\idk`, `\dgt`)
- reflecting on what you learned
- asking a question at the end of class

### 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

**Tip**

On prismia questions, I will generally give a “Last chance to get an answer in” warning before I resume instruction. If you do not respond at all too many times, we will ask you to follow the makeup procedure instead of the In Class proccedure for your experience badge.

To be sure that your response rate is good, if you are paying attention, but do not have an answer you can use one of the following special commands in prismia:

- `\idk`: “I am paying attention, but do not know how to answer this”
- `\dgt`: “I am paying attention, not really confused, but ran out of time trying to figure out the answer”

you can send these as plain text by pressing `enter` (not Mac) or `return` (on Mac) to send right away or have them render to emoji by pressing `tab`

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.

**Do you earn badges for prepare for class?**

No, prepare for class tasks are folded into your experience badges.

**What do you do when you miss class?**

Read the notes, follow along, and produce and experience report or attend office hours.

**What if I have no questions?**

Learning to ask questions is important. Your questions can be clarifying (eg because you misunderstood something) or show that you understand what we covered well enough to think of hypothetical scenarios or options or what might come next. Basically, focused curiosity.

## Lab Checkouts

## Review and Practice Badges

The tasks for these badges will be defined at the bottom of the notes for each class session *and* aggregated to badge-type specific pages on the left hand side fo the course website.

You can earn review and practice badges by:

- creating an issue for the badge you plan to work on
- completing the tasks

Skip to main content

- 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.

You can earn an explore badge by:

- creating an issue proposing your idea (consider this ~15 min of work or less)
- adjusting your idea until given the proceed label
- completing your exploration
- submitting it as a PR
- making any requested changes
- merging the PR after approval

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). 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.

> **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.

**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.

You can earn a build badge by:

- creating an issue proposing your idea and iterating until it is given the “proceed” label
- providing updates on your progress
- completing the build
- submitting a summary report as a PR linked to your proposal issue
- making any requested changes
- merging the PR after approval

**You should create one PR per badge**

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 compelte 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 in good healthy 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`.

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.

Skip to main content

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 impactful**

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 is the best way to help you learn.

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

**IDEA**

If the course response rate on the IDEA survey is about 75%, 🎁 will be applicable to final grading. **this includes the requirement of the student to reply**

## 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.

**What do you think?**

share your thoughts on this option in the discussions for the class and then

Skip to main content

# 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 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.

Skip to main content

# 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 or email address associated with a commit 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

> **Tip**
>
> Assignments are tested against LLMs and designed so that they require you to learn and that an LLM answer will be low quality and not earn credit.

All of your work must reflect your own thinking and understanding. The written work in English 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 incuding, IDE-integrated GitHub co-pilot (or similar, IDE embedded tool) 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.

Skip to main content

If work is suspected to be the result of inappropriate collaboration or AI use, 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 according to the guidelines below per badge type (you cannot treat the plagiarized badge as skipped). If you are found to have violated the policy a second time, then no further work will be graded for the remainder of the semester.

If you are found to submit work that is not your own for a *review or practice* badge, the review and practice 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.

## 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.

**You do not need to notify me in advance.**

For short absences (1-2 classes) can follow the makeup procedures.

For this, email 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. In your plan, include what class sessions you missed by date.

## 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

Skip to main content

# 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

**Warning**

For the first week they will be made by BrightSpace too, but after that, all course activities will be only on GitHub.

**Sign up to watch**

Watch the repo and then, after the first class, claim a community badge for doing so, using a link to these instructions as the “contribution” like follows.

```
- [watched the repo as per announcements](https://introcompsys.github.io/spring2023/syllabus/comm
```

put this on a branch called `watch_community_badge` and title your PR “Community-Watch”

# Help Hours

| Day | Time | Location | Host |
|---|---|---|---|
| Monday | 4:45-6:30 | 052 Tyler | Trevor |
| Tuesday | 3:00-4:00pm | 134 Tyler | Dr. Brown |
| Tuesday | 11-12:30 | 052 Tyler | Trevor |
| Tuesday | 2:00 - 4:00 | Zoom | Marcin |
| Wednesday | 2:00 - 4:00 | 049 Tyler | Gyanko |
| Thursday | 11-12:30 | 052 Tyler | Trevor |
| Friday | 2:00 - 4:00 | 049 Tyler | Gyanko |
| Friday | see GH org site | Zoom | 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, make sure that you have completed Lab 0.

Skip to main content

# 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 isses

- 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
- 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