CS 105: Programming Languages

Fall 2023

https://www.cs.tufts.edu/comp/105

Class Sessions	Monday, Wednesday 10:30am-11:45am
	Cabot Center, ASEAN Auditorium
Instructors	Richard Townsend, richard@cs.tufts.edu
	Milod Kazerounian, milod.kazerounian@tufts.edu
Richard's Office Hours (Cummings 440A)	Tuesdays 3:00pm-4:00pm
	Thursdays 11:00am-12:00pm
	(also available by appointment)
Milod's Office Hours (Cummings 469)	Tuesdays 1:00pm-2:30pm (in-person)
	Thursdays 1:00pm-2:30pm **(on zoom:** https://tufts.zoom.us/my/milod)
Final Exam	Monday, December 18 from 3:30pm-5:30pm

Course Overview

Summary

CS 105 introduces you—through extensive practice—to ideas and techniques that are found everywhere in today's programming languages. You will learn high-level, flexible programming skills that are applicable to older languages, popular modern languages, and even languages that don't yet exist. No matter what language you work in, when you finish 105, you'll be writing more powerful programs using less code. At the same time, you will learn the mathematical foundations needed to talk precisely about languages and programs: abstract syntax, formal semantics, and type systems.

You will explore and apply programming language concepts through multiple case studies, conducted using (mostly) tiny languages that are designed to help you learn. In any given case study, you may act as a practitioner (by writing code in a language), as an implementor (by working on an interpreter for the language), as a designer (by inventing semantics for a related language), or as a scholar (by proving mathematical properties of the language).

Logistics

The main point of access for the course is the website (the link is at the top of this document): it provides all the slides, readings, homeworks, solutions, and recitation materials.

Here is the weekly flow of the course:

1. Come to class sessions: take notes, participate in partner-based activities, and ask questions! This will ensure you're prepared for assignments and recitations. **Recitations will not be useful to you if you haven't come to class first.**

2. Read over the homework spec *before* attending recitation. Ask questions about it on Piazza and in office hours (OH), or talk about it with your classmates. Start the assigned reading.

3. Go to recitation, where you will get supervised practice working on problems that resemble the homework problems.

4. Work on your assignment, first by doing the reading and comprehension questions and then by completing the programming and/or proof problems. Make sure you spread this out so you have breaks between your 105 work. Post on piazza or come to OH if you need assistance.

If you feel like you're falling behind or are overwhelmed with 105, please reach out to a member of the course staff; we want to help you!!!

Prerequisites

- CS 15 Data Structures: You need substantial programming experience to keep up with the homework, particularly working with dynamically allocated (i.e., heap-allocated) data structures and recursive functions. We will also be referencing many classic data structures (stacks, queues, lists, trees) and algorithms (sorts, graph algorithms) throughout the course.

- CS/MATH 61 Discrete Math: You need some experience proving theorems, especially by induction. You should also be comfortable reading and writing formal mathematical notation.

Course Goals

By the end of this course, students should be able to

1. Design code using algebraic laws.

2. Read and write code that uses functional programming techniques.

3. Recognize the merits of polymorphism, type checking, and type inference.

4. Use functional and object-oriented language features to hide implementation details.

5. Understand precise specifications of how programming languages work.

6. Mathematically prove the behavior of programs written in a given language.

Technical Resources

Office Hours

This course is challenging, but we want to help you succeed! If you need help understanding a concept or tackling an assignment, or dealing with a pernicious bug, you can participate in our TA office hours sessions. You can also attend an instructor's office hours for any of these reasons, to discuss any issues you're having with the course, or just to say "hi!"

TA office hours will take place in the middle of the main hallway on the 3rd floor of the Cummings Center (near the stairs). When you want help, write your name and your place in the queue (as a number next to your name) on the whiteboard. TAs will prioritize groups of students who want

to talk about similar issues; note that groups cannot look at any students' code, algebraic laws, or function contracts. If you need help with algebraic laws or code that you've written or want to write, you must request private help.
Office hour assistance is meant to help you along or provide a nudge in the right direction; we expect that you will try to grapple with your issue yourself before asking for help. To help you follow this practice, you are expected to provide a specific topic or question to receive help. Here are some *poor* examples of specific topics/questions: "why doesn't my program work?", "debugging", "can you explain impcore?". Here are *much better* examples: "why does my function produce a number when it should produce a list?", "debugging a syntax error", "how do I call a function in impcore?".

The full office hours schedule is posted and kept up to date as a pinned Piazza post; make sure you check it before trekking to Cummings!

Piazza

We will be using Piazza for class discussion; you can get to the 105 page via this link or on the Home page of the course website. **Piazza will host all of our major course announcements; it is your responsibility to check in regularly to avoid missing any important information.**

In general, you should post all questions related to the course on Piazza; post publicly if at all possible, as other students may have similar questions or be able to help you faster than the course staff. Please post privately to the course staff if your question reveals individual work (e.g., algebraic laws or code) or concerns your grades.

Textbooks

This course has two required textbooks:

- *Programming Languages: Build, Prove, and Compare.* Norman Ramsey. Tufts University, Fall 2022.

  You must have this specific edition; if you're looking at an old version it may no longer align with this semester's assignments! The correct edition can be purchased from the Tufts Bookstore. If paying for this book is a financial hardship or you have any other questions about obtaining the textbook, please contact Donna Cirelli at dcirelli@tufts.edu (she's great).

- *Seven Lessons in Program Design.* Norman Ramsey.

  This short booklet is linked from our course website (at the bottom of the home page).

There is also one optional textbook that is useful for the few ML assignments we have:

- *Elements of ML Programming, ML97 Edition.* Jeffrey Ullman. Pearson, 1997.

Tentative Schedule

The tentative schedule for the course is displayed in the calendar below; topics and assignments are subject to change, and may be updated as the semester progresses (you will be informed of any changes as soon as they happen). For each week, the calendar displays the topic of Monday and Wednesday's lectures, the homework due that week (typically due on Tuesdays with a few exceptions; check the more specific schedule on the course website), and the recitation topic. Note that students in Friday recitations will have their recitation on a Tuesday (Fake Friday) during the week of Nov 5. The course website's schedule and Gradescope both provide the specific dates and deadlines for each assignment.

Week	**Monday**	**Homework**	**Wednesday**	**Recitation**
Sep 3	*(No Class)*	—	Introduction; Impcore	Impcore
Sep 10	Operational Semantics	Impcore	Operational Semantics	Opsem
Sep 17	Scheme I	Opsem	Scheme II	Scheme
Sep 24	Scheme II	Scheme	Scheme III	HOFs
Oct 1	Scheme III	HOFs	ML	Continuations
Oct 8	*(No Class: School holiday)*	Continuations	ML	*(No Recitation)*
Oct 15	ML	—	Type Systems	ML
Oct 22	Type Systems	ML	Type Systems	Type Systems
Oct 29	Type Systems	Type Systems	Type Inference	*(No Recitation)*
Nov 5	Type Inference	—	Type Inference	Type Inference
Nov 12	Modules	Type Inference	*(Lecture TBA)*	Modules
Nov 19	Lambda Calculus	Modules	(No Class)	(No Recitation)
Nov 26	Smalltalk	—	Smalltalk	Smalltalk I
Dec 3	Smalltalk	—	Course Wrap-Up	Smalltalk II
Dec 10	*(Lecture TBA)*	Smalltalk	*(Classes end)*	*(Classes end)*

Assessments and Grading

Your course grade will be produced as a weighted sum of your scores in four categories:

1. Recitations (5%)
2. Homework: Comprehension Questions (10%)
3. Homework: Programming and Proof Problems (70%)
4. Final Exam (15%)

All grades will be posted on Gradescope for students to review.

Although your ultimate course grade will be a conventional letter, much of your specific work will be graded on a scale of No Credit, Poor, Fair, Good, Very Good, and Excellent. These grades correspond to numeric values of 0, 65, 75, 85, 95, and 100, respectively. For example, if an assignment has 3 equally weighted components and you received a Good on two parts and a Very Good on the third, your grade for that assignment would be approximately .33 * 85 + .33 * 85 + .33 * 95 = 87%. In a typical class, a consistent record of Very Good work will lead to a course grade in the A range. Work rated Good corresponds to a wide range of passing grades centered around B. Work rated Fair will lead to low but satisfactory course grades around a C; if a significant fraction of your work is Poor, you can expect an unsatisfactory grade.

Recitations

Recitations are graded as Attended or Not Attended (a recitation TA will take attendance in each session); your final recitation grade is the percentage of recitations you attended. We drop your two lowest recitation grades, effectively letting you miss two recitations at no penalty. There is no such thing as an "excused absence" from recitation, so if you need to miss one just let your recitation leader know. If you need to miss more than two recitations due to a personal emergency, please explain the situation to your academic dean and ask them to contact a course instructor.

Homework

In general, homeworks will go out every Wednesday and be due at 11:59:00PM EDT the following Tuesday (there will be a few larger assignments in the second half of the course that are exceptions to this rule). Each homework will have two components:

1. Your responses to a set of reading comprehension questions (CQs), which will be submitted as a .txt file.
2. A programming and proof component, which will be submitted either via our course-specific submission scripts or as a **.pdf** file submitted to Gradescope.

To receive a grade, all homework files must be submitted via the specific instructions that will be provided on each assignment. No submissions will be accepted via email or other means. To avoid missing the strict 11:59:00PM EDT deadline, you should submit whatever you have a bit earlier (or much earlier!) even if incomplete; you may submit as many times as you wish before the deadline. We will always grade the latest submission.

When you use a submission script to submit any pdf files, they will be emailed back to you so you can verify that they are viewable. **It is your responsibility to check this email and ensure that you submitted the correct, viewable pdf. If you forget to check and the submitted pdf is "the wrong one" or not viewable, no regrade will be considered.** Avoid using VSCode to upload any pdfs to the departmental servers; it is known to corrupt pdf files.

Comprehension Question Grades A set of comprehension questions (CQs) is associated with each homework assignment. Your answers to these questions are graded based on the number of questions answered and the correctness of each answer, roughly following this scale:

- **Excellent**: All answers are completely correct.
- **Very good**: All answers are mostly correct.

- **Good**: A majority of answers are mostly correct.
- **Fair**: Most answers are incorrect, but all were reasonably attempted.
- **Poor**: Most answers are incorrect or blank.
- **No Credit**: No answers are provided.

Programming and Proof Problem Grades The programming and proof problems on each assignment will be graded both for correctness (e.g., Do your programs pass our automated tests? Are your theoretical proofs and rules accurate and complete?) and for structure and organization (e.g., Does your code follow our course coding standards? Are your algebraic laws accurate and complete?). Correctness is typically evaluated by automated testing scripts that list which tests you passed or failed and why. Structure and organization is evaluated manually by our course staff. Both of these components are graded with a coarse five-point scale:

- **Excellent**: Your submission is outstanding in all respects.
- **Very Good**: Your submission does everything asked for, and does it well. There may be something small that could be improved.
- **Good**: Your submission demonstrates quality and significant learning.
- **Fair**: Your submission is quite lacking in one or more aspects; key issues need to be addressed. This is the lowest satisfactory grade.
- **Poor**: Your submission shows little evidence of effort or has other serious deficiencies. This is an unsatisfactory grade.
- **No Credit**: No submission provided OR the submission exhibits plagiarism OR the submission violates a rule that the homework indicated would result in no credit.

Late Policy

Each student is automatically issued **seven "late tokens"** to be used on assignments. It is your responsibility to track how many late days you've used over the semester. By expending a late token, you get a 24-hour extension on an assignment. This extension happens automatically: when you turn in any piece of work past the deadline, the course software charges you one late token. Expenditure of late tokens is governed by these rules:

- At most **one** late token can be used on a single assignment. **No submission will be accepted more than one day late.**
- Once you are out of late tokens, late homework will no longer be accepted and will receive a score of No Credit.
- If you are pair programming with another student (on an assignment that allows it), submitting late will expend a token from **both** partners. Furthermore, both partners must have enough late tokens to receive credit for a late submission. For example, if one student in a group has a late token but the other does not, both students will receive No Credit if the submission is late.

You should think of late tokens as extensions you have been granted ahead of time and use them when you might have otherwise tried to ask for an extension.

If you experience an extraordinary difficulty, such as serious illness, family emergencies, or other extraordinary unpleasant events, your first step should be to contact your advising dean as soon as you can: explain the situation to them and ask them to contact a course instructor. Your dean will work with the instructor to make appropriate arrangements. **IMPORTANT: The earlier you notify your dean, the more flexibility the course staff will have to make appropriate arrangements.**

Regrades and Grade Explanations

If you want to request a regrade for an objective error on our part, you must submit a private request to the Instructors via Piazza; provide your UTLN and explain the error. The deadline for these requests is **one week** from the time the grades were posted for a given assignment; no exceptions to this policy will be made. A regrade request may or may not result in a new grade being assigned.

If a regrade request requires the course staff to make a manual modification to your submission, your grade for the component being regraded will be capped at a "Good" as a penalty. **IMPORTANT: No regrades are considered for incorrectly named functions on autograded assignments.** The names of these functions will always be provided to you, and it is your responsibility to follow our assignment's directions.

You are always welcome to ask for an explanation of the grade you received. We want to help you understand how your work was evaluated and how you can continually improve in the course!

Collaboration and Academic Honesty

If you have **any** questions about the below policies or a specific situation dealing with academic integrity, do not hesitate to ask a course instructor. All students are expected to read and adhere to the Tufts Academic Integrity Policy.

Collaboration

In general, you are encouraged to discuss the lecture content, reading material, and what's being asked of you on assignments with your classmates. However, **you must not discuss anything that you produce for an assignment with classmates in any form (e.g., English, pseudocode, pictures, etc.).** Examples include proofs, code, algebraic laws, and function contracts. You may not show your work at this level to other students nor may you look at others' work at this level. This also applies to testing and debugging: you may not test or debug another student's code or let them test or debug your code. Due to this policy, any Piazza post that discloses your work on an assignment must be posted privately.

Lifting or looking at partial or complete solutions from anyone (classmates, online sources, strangers) is completely prohibited. You may not work on the specific problems with anyone other than TAs or a course instructor, and you should not use the internet in any capacity to help solve any specific problem. No homework problems, student solutions, or instructor-provided solutions should be posted online in any capacity (except as a submission to Canvas); **this means that you are**

prohibited from posting any of your work for 105 in a public Github repository.

If an assignment contains a *pair programming* component and you choose to pair program, you are welcome to discuss/view any part of that component with your partner, but not any other students.

Policies for Students Retaking CS 105

If you are repeating the course or any part of it (e.g., you withdrew or dropped after doing one or more assignments), you must let a course instructor know at the beginning of the semester. In this case, you are expected to abide by the following policies:

- There is no acceptable use of instructor-provided solutions from prior semesters. If you have kept any such solutions, destroy them.
- If you have written your own solutions for past semesters, it is acceptable to consult them for ideas, and it is acceptable to submit parts verbatim. Such use must be explicitly acknowledged in a README.
- In every homework, your README file must note what work is new, what work is based on work from a prior semester, and what work is submitted verbatim from a prior semester. Even if nothing is from a prior semester, your README file must disclose this information.
- In every homework, you must cite collaboration with every partner with whom you worked on the assignment in a past semester—even if none of that partner work survives.

Academic Honesty

Any violation of the above policies will be considered cheating, and all students suspected of being involved in any capacity will be forwarded directly to the Office of Student Affairs, who will investigate the case independently. Their sanctions range from horrible to inconceivably horrible. It's not worth it.

Inclusivity, Accessibility, and Additional Help

This course strives for inclusion of all participants, regardless of personal identity (gender, race, sexual orientation, religion, etc.), socio-economic background, disabilities, or neurodivergence. In the classroom and our discussion forums, everyone is expected to treat everyone else with dignity and respect. If you feel unwelcome or mistreated for any reason by either another student, a TA, or the course itself, please let an appropriate member of the teaching staff know so we can work to make things better.

CS 105 can be especially difficult for first-generation college students and for members of historically underrepresented groups in computing, who may not have the family or social support that helps them develop their skills in "how to be a college student." If you are a student in either of these categories, you are strongly encouraged to meet with a course instructor or contact one early in the term to talk about your support system.

Accommodations for Students with Disabilities

Tufts University values the diversity of our body of students, staff, and faculty and recognizes the important contribution each student makes to our unique community. Tufts is committed to providing equal access and support to all qualified students through the provision of reasonable accommodations so that each student may fully participate in the Tufts experience. If you have a disability that requires reasonable accommodations, please contact the StAAR Center at StaarCenter@tufts.edu or head to the StAAR Center website to make an appointment with an accessibility representative to determine appropriate accommodations. **Please be aware that accommodations cannot be enacted retroactively; all accomodation notes must be provided to the instructor wihtin one week of the note being written to guarantee consideration.**

Academic Support at the StAAR Center

The StAAR Center offers a variety of resources to all students (both undergraduate and graduate) in the Schools of Arts and Sciences, and Engineering, the SMFA, and The Fletcher School; services are free to all enrolled students. Students may make an appointment to work on any writing-related project or assignment, attend subject tutoring in a variety of disciplines, or meet with an academic coach to hone fundamental academic skills like time management or overcoming procrastination. Students can make an appointment for any of these services by visiting tutorfinder.studentservices.tufts.edu, or by visiting students.tufts.edu/staar-center.

Religious Holy Days

We will reasonably accommodate any student who, for reasons of observing religious holy days, will be absent from recitation or experience any hardship in the completion of their work during the holy days. Please contact the instructor in advance of the holy day if you will miss recitation or if you need any additional accommodation (such as for assignments); reasonable accommodations will be allowed at no penalty **only if the instructor is notified well in advance**.

Acknowledgements

This syllabus (and the entire course) uses many ideas and materials developed by previous instructors of CS 105, including Norman Ramsey, Kathleen Fisher, and Jeff Foster. Much thanks!