# Course Manual

CAS CS 320: *Concepts of Programming Languages*

Boston University

Fall 2025

# Contents

| | |
|---|---|
| Course Code | `CAS CS 320` |
| Course Title | Concepts of Programming Languages |
| Semester | Fall 2025 |
| Instructors | Ankush Das and Nathan Mull |
| Teaching Fellows | Sam Silverman and Uğur Yavuz |
| Course Assistants | Miranda Quimbar and Shiyun Yang |
| Meeting Times | Tuesday and Thursday, 3:30PM-4:45PM (A1) |
| | Tuesday and Thursday, 5:00PM-6:15PM (A2) |
| Meeting Location | CAS 211 |
| Midterm Date | October 16 (during lecture) |
| 20% | Assignments |
| 16% | Quizzes |
| 24% | Mini-projects |
| 20% | Midterm Exam |
| 20% | Final Exam |

Figure 1: Course overview

**Abstract**

This is a manual[1] for the course `CAS CS 320`: *Concepts of Programming Languages*, which will be taught at Boston University during the Fall 2025 semester. It contains a general overview of the course and its policies. It does *not* contain specifics about the material being covered in the course; this appears on the course webpage. Figure 1 contains an overview of the course.

[1] Which takes the place of a syllabus.

# Week 0 To-Do list

You should complete the following items within the first 48 hours of the start of the semester. Please reach out if you have concerns about any of the items listed.

- ☐ Verify that you have ready access to a laptop computer during the semester
- ☐ Verify that you know where the lecture is held
- ☐ Verify that you know where the discussion section in which you're registered is held
- ☐ Join Piazza with the following sign-up link
- ☐ Join Gradescope with the following sign-up link (entry code: VWP43X)
- ☐ Familiarize yourself with the course webpage
- ☐ *(Optional)* Add the Piazza, Gradescope, and course webpage as bookmarks in your Internet browser
- ☐ Review the course calendar and determine which office hours you're able to attend
- ☐ *(Optional)* Add the course calendar to your own calendar
- ☐ Read this manual in its entirety
- ☐ Submit the assignment on Gradescope confirming that you've read this manual
- ☐ If necessary, review material from `CS210` in order to be comfortable in the terminal

# General Information

`CAS CS 320`: *Concepts of Programming Languages* is a course *about* programming languages, particularly the **design** and **implementation** of programming languages. In this course, we take up the programming language as an object of formal study. This course is *not* about how to program, though the principles we cover are generally useful for writing and reasoning about programs.

The first part of the course is on functional programming in OCaml, based on `CS3110`: *Data Structures and Functional Programming* at Cornell University. The topics include functional design, algebraic data types, higher-order functions, polymorphism, functional data structures, and modular programming. It's during this part that we learn to *think* functionally, to view programs not as sequences of commands manipulating global state, but as compositions of functions that deconstruct and reorient data. We also introduce in this part of the course fundamental concepts from the theory of programming languages—like inference systems, typing rules, semantic rules, and derivations—with an eye towards how these concepts apply to OCaml in particular.

In the second part of the course we implement several interpreters for fragments of OCaml; that is, we write OCaml programs that *execute* other OCaml programs. The topics covered include formal grammar, parsing, operational semantics, variable scope and binding, closures, type checking, type safety, and type inference. By the end of the course, you'll be able to execute some of the simpler programs we wrote in the first part of the course using your own interpreter. You'll also be able to reason formally about what a programming language is, and what makes some programming languages good and others not-so-much.

## 2.1 Prerequisites

The formal prerequisites for this course are:

- `CAS CS 111` & `112`: *Introduction to Computer Science*
- `CAS CS 131`: *Combinatoric Structures*
- `CAS CS 210`: *Computer Systems*

You'll get the most out of this course (and you'll have the best time of it) if you've completed the 200 level major requirements for the computer science degree at BU. Experience with a high-level programming language (like Python or Java) is a must. The dependence on `CS131` is a level of mathematical maturity and exposure to the principle of induction. We don't depend explicitly on the material of `CS210`, but it will be useful in appreciating the second half of the course. We also assume general comfort in the terminal. If you don't have experience with this, then you'll be have to self-learn it within the first week of the course.

## 2.2 Learning Objectives

From this course we hope that you will:

- Learn the rudiments of OCaml, a functional programming language which is likely very different from other languages you've learned. In particular, we'd like you learn to "think functionally" and understand the benefits of doing so.
- Learn to read and write formal specifications for programming languages.
- Gain an understanding of the theoretical basis for what makes a "good" programming language.
- Gain an appreciation for what goes into the development of the programming languages we use everyday, particularly by implementing interpreters for a collection of OCaml-like languages.

`CS320` fulfills a single unit in the BU Hub area **Creativity/Innovation**. This deserves some explanation. Programming language design is a perhaps surprisingly creative process, in which personal aesthetics play a large role in what features are included (or not) in a language. A fair amount of creativity also goes into discovering new features that aid the safety, productivity, or ergonomics of a new language.[1] This is in part to say that what we cover is based on our notion of what a "good" programming language is, which won't necessarily align with your own views or the views of other programming language designers. This also must be understood in spite of the fact that much of what we do in the course will not feel very creative. All I can say on this point: you have to know what came first before you can change the state-of-the-art.

One more point I'd like to make. It's no secret that `CS320` is not everyone's first choice for a course to take in our department. OCaml is not a terribly popular language, and programming language design doesn't have the same hype as machine learning or data science (or even much of what is considered "systems" programming). Now, I wouldn't go as far as saying that OCaml and programming languages design are "on the rise," but there are some interesting trends worth noting.

- OCaml is being heavily used in and developed by the trading firm Jane Street, and is being used to a lesser extent by companies like Bloomberg, Docker, and Facebook, among others.
- OCaml won the ACM SIGPLAN's 2023 Programming Languages Software Award.[2]
- In LinkedIn's ranking of the 50 top colleges in the US for long-term career success, one of the "most notable skills" offered by BU (university-wide) is OCaml.[3]
- I've been told that OCaml may become a language which is used in competitive programming.
- Rust (a fairly new systems programming language with a lot of hype) was originally written in OCaml and borrows many of its functional features. In the other direction, Jane Street recently came out with OxCaml, an extension to OCaml with Rust-like features.
- Rust itself is an fascinating contemporary example of programming language design having a major impact on the world of software engineering.
- Domain specific language and compilers are becoming more common tools for dealing with bottlenecks in high-level languages, in areas as wide-ranging as machine learning, networks, and scientific computing.

Just a couple things to think about as you're determining what skills are worth having in your toolbox...

[1] Rust is a great example of a modern language with a fair amount of creativity behind its design.
[2] `https://ocaml.org/news/sigplan_announcement`
[3] `https://www.linkedin.com/pulse/linkedin-top-colleges-2025-50-best-long-term-career-success-kritf/`

## 2.3 Course Structure

### Lectures

We hold lectures each week on Tuesday and Thursdays (see the registrar and the course webpage for details). During lecture, we cover the material that is presented in the reading, do live coding examples, and provide practice problems. The material used in the lecture is made available on the course webpage before the lecture meeting. Barring technical difficulties, recordings of the lecture will be made available.

We won't take attendance during lecture, but it is highly recommended that you attend, and refer you to the BU Attendance policy. You'll be expected to participate in lectures by working on practice problems and occasionally discussing topics with the people sitting around you.

### Discussion Sections

We hold discussion sections each week on Wednesdays (see the registrar and the course webpage for details). The will have programming tasks and worksheets (see the course webpage for details).

We won't take attendance during discussion sections, but it is highly recommended that you attend. Likewise, you *must* take the quizzes in the discussion section in which you're registered if you want to receive credit for them.

## 2.4 Resources

### Material

The first half of the course uses the textbook OCaml Programming: Correct + Efficient + Beautiful. All other course material will be made available on the course GitHub repository and on the course webpage. If you are unfamiliar with git and GitHub, see the GitHub documentation for information and tutorials. Please check the course webpage and repository frequently. We'll also be testing out some new notes for this course. These will be made available on the course webpage.

### Programming

The programming in this course is done in OCaml. You're required to set this up on you personal machine or on a machine that you'll have access to throughout the semester. You'll have an opportunity to do this in your first discussion section. Please attend office hours and use Piazza if you need help troubleshooting. If you're are worried about access to technology, please contact me as soon as possible and we can see what we can do (though I cannot make any guarantees).

### Course Communication

Course announcements and discussions will happen on Piazza. If you're unfamiliar with Piazza, see their support page for information and tutorials. Some policies regarding the use of Piazza:

- *Don't ask homework questions directly.* Formulate a question which will aid in your understanding, and will hopefully help others as well.
- *Don't give homework solutions directly.*
- *Piazza is as useful as it is active.* Teaching fellows and course assistants will be answering questions on Piazza, but don't hesitate to answer questions yourself.

Make sure to set notifications correctly so you can keep up with updates regarding the course. “I didn’t see the Piazza post about it” is never a valid excuse for missing a piece of information.

## Submission

We’ll be using Gradescope for assignment submissions. If you are unfamiliar with Gradescope, see their Get Started page for information and tutorials.

# Evaluation

The grading breakdown for this course is given in Figure 3.1. The sites of evaluation are detailed in the following sections. Your raw percentage will be determined according to this breakdown and your final letter grade is guaranteed to be at least what is determined by Wheelock College's Grading Scale.[1] But, to borrow a phrase from Professor Mark Bun: "to correct for the possibility of [quizzes] and exams being more difficult than anticipated, letter grades may be (significantly) increased above these guarantees." Specifically, we may retroactively curve exam and quiz grades using a linear scale.[2]

## 3.1 Assignments

Assignments are released (roughly) weekly on Thursdays during the first half of the semester and are due a week later on the following Thursday by 8:00PM. See the calendar on the course webpage for details. Assignments consist of written problems that are to be submitted as a pdf file via Gradescope, as well as programming problems that will be autograded. There are 6 assignments total. We drop your lowest assignment grade, so only 5 assignments will count towards your final grade in the course. We don't accept late assignments under any circumstances.

You'll notice that, despite the fact that there are many assignments, they account for a very small portion of your final grade. You should think of the assignments in this course as *accountability checks* in that they require to engage with the material each week. But the more effort you put into learning and internalizing the material in the assignments, the more successful you'll be in the other evaluation sites of the course, like the quizzes and exams.[3]

[1] Formally we're part of the College of Arts and Sciences (CAS), but this grading scale is standard.

[2] See this article for details if you're interested.

[3] We understand that this approach to evaluation will not be universally liked, and that in-person evaluation can be challenging. We take this under careful consideration when we calibrate the difficulty of the material and the workload we expect.

| | |
|---|---|
| 20% | Assignments (6 total, 1 dropped) |
| 16% | Quizzes (2 total, 1 dropped) |
| 24% | Mini-projects (3 total) |
| 20% | Midterm Exam |
| 20% | Final Exam |

Figure 3.1: Grade breakdown

## 3.2 Quizzes

There will be two quizzes during the semester held during lecture. They will consist of problems *very* similar to the problems that have appeared on previously submitted assignments. If you've completed the assignments and practiced the concepts therein, you're expected to do well on the quizzes. There are 2 quizzes total. We drop your lowest quiz grade, effectively taking the maximum of the two scores. However, **you must take both quizzes** in order for the drop to take effect. If you do not take a quiz, you are giving up 8% of your final grade.

## 3.3 Mini-Projects

Mini-projects are released (roughly) every other week on Thursdays in the second half of the course, and are due two weeks later on Thursday by 8:00PM. See the calendar on the course webpage for details. Each mini-project requires you to build an interpreter. Each interpreter will be slightly more complex than the last. You should think of these as roughly the same amount of work as two assignments, but with a single large programming task (building an interpreter) instead of small programming tasks. Mini-projects will be accompanied with a check-in, which is another form of *accountability check* to verify you're making progress on the project. There are 3 mini-projects total. **It's not possible to drop a mini-project.**

## 3.4 Midterm Exam

The midterm exam will be held on October 16 during lecture. It will be a closed-note written exam. It is meant to verify that you're prepared to program more heavily in OCaml and to go further in depth into programming language concepts during the second half of the course.

## 3.5 Final Exam

The date of the final exam will be determined later in the semester. It will be a closed-note written exam. It is meant to verify that you've internalized the basic concepts of the course, and can also apply them to solve novel problems.

# Policies

There are a number of policies associated with this course, some specific to the course and others which hold more generally in the university. These policies are detailed below in the following sections.

## 4.1 Diversity Statement

Our aim is to present material in a way that respects the diversity of the student body. If we fail to do this, please make us aware. Any suggestions are welcome and appreciated. We also expect students to appreciate and respect the unique opportunity they have to participate in a diverse student body like ours.

## 4.2 Disability Statement

If you require disability accommodations, please contact us as soon as possible. You should provide us with the appropriate documentation, available through the Disability and Access Services. In order to receive accommodations, you *must* be in contact with us.

If there's a policy that we're failing to comply with, please reach out with suggestions. And if you'd like accommodations that aren't covered by existing services or policies, feel free to contact us and we can see what we can do. We want everyone to feel able to fully participate in the course.

## 4.3 Sexual Misconduct

Please read the Sexual Misconduct Policy and review the entire page for information on talking to someone confidentially about experiences of sexual misconduct, filing a report, and any other relevant information. Above all, you should feel safe, and able to be productive. If this is not the case, please reach out to us or someone else immediately.

The members of the course staff are considered "mandated reporters" and are required to report cases of sexual misconduct. Therefore, **we cannot guarantee the confidentiality of a report**. We must provide our Title IX coordinator with relevant details such as the names of those involved in the incident. The university will consider a request for confidentiality and respect it to the extent possible.

With that in mind, if you come to any of us with questions or concerns, we will handle the situation to the best of our ability and connect you with available resources.

## 4.4 Academic Integrity

Please read the Academic Conduct Code and review the entire page for information about what constitutes academic dishonesty and what penalties arise as a result of violations of this code. This is taken very

seriously at BU and we take it seriously in this courses. There are a couple policies about which we'll be strict:

- You must submit your own work for all assignments and mini-projects. Submitting the same file as another student, or something notably similar (e.g., identical wording or code in large parts of the solution) is considered academic misconduct and will be handled accordingly.
- Copying or information sharing regarding in-class evaluations like quizzes and exams is considered academic misconduct and will be handled accordingly.

If you work with others, consult materials found on the Internet, or use an AI assistant, you should cite your sources. This is a useful skill in any setting, and so we recommend being as conservative as possible regarding citations. In any assignment, these citations should appear next to every corresponding problem (in comments if the submission is code). Some examples:

- I discussed problem 1 and 2 with Leah Smith. She helped me understand X and Y aspects of the problem.
- I saw the stack overflow post stackoverflow.com/questions/6681284/python-numpy-arrays which informed my solution.
- I helped Zihan Guo with problem 4. I told them to try using X.
- I asked ChatGPT "what's the largest eigenvalue of this matrix?"

When in doubt, err on the side of longer, more descriptive citations. We do not consider missing or poor citations is a direct act of academic misconduct, but we will consider this grounds for further investigation in suspicious cases. Above all, use your best judgment and remember:

- We care about your success in this course. We provide a number of avenues to ask for help, please use them.
- You will have to answer questions on quizzes and exams without external aids (and in interviews when you apply for a job).
- If you don't know how to start thinking about a problem, it's okay to ask for pointers in office hours and on Piazza.
- We have safeguards (like dropped homework assignments) in the case you are unable to complete an assignment. In other words, don't submit someone else's work when you can drop an assignment.

## 4.5 Generative AI

The problem of generative AI in higher education will likely occupy us for the next decade or so. The role of these tools our lives is still an open question, one with many possible answers. But these tools exists, and the university, for better or for worse, has made them more accessible with the introduction of TerrierGPT, to which all students of the university have access. As such, all courses (including ours) are reviewing their policies. Keep in mind that this is all one big experiment. We don't know if our policy makes sense in the long term (or even now). But it's our attempt to come to terms with the appearance of these tools in our courses.

The policy this semester: **The use of generative AI to produce solutions to assignments and mini-project is prohibited.** You are allowed to use these tools to aid your learning. We understand that this is

difficult to enforce, much of this will be on the honor systems. In an lower-division course, we might have a different policy, but a this point in your academic career, we expect that you are able to make the correct judgments as to the use of generative AI. Also, as you might have noted, we've re-weighted evaluation sites in order to put a greater emphasis on in-class closed-book evaluation.

An obligatory concluding remark: we know that existing models can solve many of the problems we will ask you. This does not negate the value in knowing how to do them without the help of external tools. To draw an imperfect analogy, we don't learn a new language in order to have memorized a vast collection of words and grammar rules, but in order to *internalize* the language, and learn how to *interact* with it and in it. This is our goal in this course and beyond. These tools can be incredibly useful in the process of learning and internalizing. But the internalizing is what we really want to achieve, so we can see a problem and can *sense* the underlying structure to which we can apply our knowledge from this course.

## 4.6 Additional Attendance Policies

As we've noted, we won't take attendance in our course. Instead, we remind you that, according to the Attendance policy at BU, you're required to attend the courses in which you're registered.

### Absence Due to Religious Observance

According to the BU policy on Absence Due to Religious Observance: you "shall be excused from any such examination or study or work requirement, and shall be provided with an opportunity to make up such examination, study, or work requirement that may have been missed because of such absence on any particular day; provided, however, that such makeup examination or work shall not create an unreasonable burden upon such school."

### Bereavement

According to the BU policy on Student Bereavement: you "should be granted up to five weekdays of bereavement leave for the death of an immediate family member." Your advisor should help you coordinate your leave.

## 4.7 Additional Grading Policies

### Regrade Requests

Regrade requests may be submitted on Gradescope for up to one week after receiving the grade for an evaluation site. Regrade requests will only be considered in the case that the grader has made a mistake in grading. Any regrade requests which solely appeal for a higher grade will not be considered.

### Grading Grievances

According to the BU policy on Grade Grievances: you may "contest a final course grade received in a unit-bearing Boston University course when that grade is alleged by the student to be arbitrary." Read the policy for more information. We recommend contacting us before submitting a formal appeal.

## Incomplete Grades

According to the BU policy on Incomplete Coursework: "An incomplete grade (I) is used only when the student has conferred with the instructor prior to the submission of grades and offered acceptable reasons for the incomplete work. An incomplete grade may be appropriate when the student has participated in and completed requirements representing a majority of the course, and circumstances prevent the student from completing remaining requirements by the conclusion of the course." In particular, **you must contact us before the last day of the semester in order to receive an incomplete grade.**

# Closing Remarks

Quite a bit goes into organizing a course (even producing this document) as well as taking a course (even reading this document). In light of my comments on citations, I'll note that much of what's in this document is based on similar documents (often taken without permission) by Mark Crovella, Mark Bun, Jonathan Appavoo, Preethi Narayanan, Ravi Chugh, Andrew McNutt, and others I may be missing. All told, we hope that most of this logistical information will be overshadowed in your memory by the concepts themselves, and that we can focus on having a good time doing math and programming!

## 5.1 Course Agreement

In addition to a manual, we also consider this document a contract. The following is what you must agree to in order to remain in this course.

> *By enrolling in this course, I am agreeing to the policies outlined in this document, and I will uphold them to the best of my ability. I will also, generally speaking, try to be a reasonable person and be nice and good and respectful to the people around me taking—and running —the course. In return, I expect a high-quality learning experience and respect from those around me taking—and running—the course.*

## 5.2 University Resources

There are quite a few BU resources, it can sometimes be overwhelming. Here's a small list of the ones we think are important. If you're struggling in this course due to personal/health conditions, we can't guarantee we can help, but if you're comfortable reaching out, feel free to send us an email and we can see if we can point you towards the correct resources. If you're not comfortable reaching out to us, that's okay too, hopefully this list can help you find what you need. Also, keep in mind you can post anonymously on Piazza if you want to ask for help without including your name.

- Disability and Access Services
- Student Health Services
- Outreach and Prevention
- Behavioral Medicine
- Survivor Support (SARP)
- Educational Resources Center
- International Students & Scholars Office