

Course Information

Overview

This page covers the course policies for CS 61A, the in-person version of the course. If you are looking for the course policies for CS W61A, the online version of the course, please see here ([/~cs61a/su19/articles/about-w61a.html](https://inst.eecs.berkeley.edu/~cs61a/su19/articles/about-w61a.html))

The CS 61 series is an introduction to computer science, with particular emphasis on software and on machines from a programmer's point of view.

1. CS 61A concentrates on the idea of abstraction, allowing the programmer to think in terms appropriate to the problem rather than in low-level operations dictated by the computer hardware.
2. CS 61B deals with the more advanced engineering aspects of software, such as constructing and analyzing large programs.
3. CS 61C focuses on machines and how they execute programs.

In CS 61A, we are interested in teaching you about programming, not about how to use one particular programming language. We consider a series of techniques for controlling program complexity, such as functional programming, data abstraction, and object-oriented programming.

CS 61A primarily uses the Python 3 programming language. Python is a popular language in both industry and academia. It is also particularly well-suited to the task of exploring the topics taught in this course. It is an open-source language developed by a large volunteer community that prides itself on the diversity of its contributors. We will also use two other languages in the latter half of the course: the Scheme programming language and the Structured Query Language (SQL).

Mastery of a particular programming language is a very useful side effect of CS 61A. However, our goal is not to dictate what language you use in your future endeavors. Instead, our hope is that once you have learned the concepts involved in programming, you will find that picking up a new programming language is but a few days' work.

Prerequisites

Math 1A is a corequisite for CS 61A. (That is, it may be taken concurrently.) Math 10A or Math 16A are also fine.

There are no formal programming-related prerequisites for admission to CS 61A, but it's not the right first course for all students. Many CS 61A students have had significant prior programming experience, including prior coursework. Some students take the course without any prior programming experience, but they typically must work substantially harder to master the material, perhaps simply because they have less practice working with programs. If you have limited prior experience and you find it challenging to complete all of the required coursework in the first three weeks, you should seriously consider taking another course first. You'll likely have a better experience taking 61A later, and you won't fall behind in any meaningful way by taking one of the alternatives below prior to taking 61A.

Alternatives

If you want to build programming experience before taking CS 61A, we recommend that you take one of these courses first. Both are offered this summer. You can always take CS 61A in a future semester.

CS 10

CS 10: The Beauty and Joy of Computing (<http://cs10.org>) provides a bird's-eye view of the field of computer science. The course teaches students how to program using Snap (based on Scratch), one of the friendliest programming languages ever invented, as well as Python, the same language used in 61A. But the course is far more than just learning to program! You'll also learn about some big ideas of computing, such as abstraction, design, recursion, concurrency, simulations, and the limits of computation. You'll also see some beautiful applications of computing that have changed the world, as well as talk about the history of computing and where it will go in the future.

Data 8

Data 8: The Foundations of Data Science (<http://data8.org/>) is an introduction to data science designed to be accessible and useful for all Berkeley students, and built for students without prior programming experience. The course teaches students to program in Python 3, but covers a much smaller subset of the language than CS 61A. Most of the course focuses on data processing and statistical techniques that are central to using computers to answer questions about the world. The overlap between Data 8 and CS 61A is small (perhaps 25%), but the programming skill you will acquire in Data 8 will help you maintain the faster pace of CS 61A.

Course Format

The course includes many events and opportunities for learning: lecture, lab, discussion, office hours, guerrilla sections, and group mentoring. We understand that everyone learns differently, so not all of these events are required. However, it is recommended that you at least try everything out to figure out what combination of these events works best for you.

Lecture

There are four 80-minute lectures per week. Slides will be posted soon after each lecture. Videos, either a screencast or a video recorded by a past instructor, will also be available. Videos are primarily available for the purpose of review, or if you have to miss a lecture or two. You shouldn't make a habit of missing live lecture, if at all possible.

This course moves very fast, so you should always watch lecture before your section that day. The TAs will assume that all of their students have watched that day's lecture.

Section

There are two labs and two discussion sections each week. These sections are run by an amazing group of teaching assistants who have been carefully selected for their ability, enthusiasm, and dedication to learning. Getting to know your TA is an excellent way to succeed in this course. Participation in lab and discussion determines your participation score for the course.

Section works best when you attend lab and discussion taught by the same TA. Due to a limitation of the campus scheduling system, we were unable to link labs and discussions, but **please try to attend the lab taught by the same TA as your discussion if at all possible.**

To find the lab section that corresponds to your discussion, take your discussion section number (ranging from 101 to 114) and subtract 90 from it (to get a number ranging from 11 to 24). Please make sure that the lab corresponding to your discussion works with your schedule, since **we'll give seating priority in lab to those enrolled in the corresponding discussion, not those enrolled in the lab section itself.** If the lab section corresponding to your discussion does not work with your schedule, please fill out this form (<https://docs.google.com/forms/d/e/1FAIpQLSeJN-A8FXllOx24y3M96nyavly599t3bnSwrUGtRYILXK2AoQ/viewform>).

Office Hours

Attending office hours is another excellent way to succeed in this course. You can ask questions about the material, receive guidance on assignments, work with peers and course staff in a small group setting, find project partners, and learn about computer science at Berkeley. We will post the office hour schedule here ([/~cs61a/su19/office-hours.html](http://~cs61a/su19/office-hours.html)).

In addition to our regular office hours, we will have a limited number of conceptual office hours. We will not take any questions about assignments at these office hours, but they will be a chance for you to ask questions about the material in lecture or discussion or to get additional practice with the concepts presented in class.

Group Mentoring Sections

Optional group mentoring sections are held each week. Each section features worksheets that review topics covered in discussion section. These sections of at most 5 or 6 students meet twice a week and are here to create a stronger feeling of community in the class and reinforce conceptual understanding of course material. These will be recurring sections which will have the same group of students, and sign-ups for these sections will open the first week of classes.

Guerrilla Sections

Attending guerrilla sections is a great way to review the entire week's worth of material. You will work in small groups to complete a worksheet, while being able to get feedback from staff. The questions in the worksheet will progress in difficulty, so this is a fantastic opportunity to build foundational understanding and work on challenging problems. We anticipate that these sections will be held on Friday from 11:00 - 1:00 PM, location TBD.

Assignments

Each week, there will be problems assigned for you to work on, most of which will involve writing, debugging, and discussing programs. These assignments come in three categories: lab exercises, homework assignments, and projects.

Labs

Lab exercises are designed to introduce a new topic. You can complete and submit these during the scheduled lab sections, or on your own time before the scheduled due date. Most students find that attending lab is much more useful than working on lab assignments independently.

Lab exercises are scored on correct completion. To receive credit, you must complete all of the problems that are not marked as optional and pass all tests.

We expect for there to be 14 lab exercises, including two ungraded exam review labs. The two lowest lab grades are dropped. As such, You will only have to complete 10 labs to get full credit for lab exercises.

Lab attendance will be tracked and worth participation credits. See Lab and Discussion Participation for more information.

Homework

Homeworks are weekly assignments meant to help you apply the concepts learned in lecture and section on more challenging problems. They will usually be released on Thursday and be due the following Tuesday night.

You are encouraged to discuss the homework with other students, as long as you write your own code and submit your own work. Finding a study group is a great idea. The purpose of homework is for you to learn the course material, not to prove that you already know it. Therefore, you can expect to receive substantial assistance from the course staff and your grade will ultimately be based on showing effort on every problem, even if you don't pass all the tests.

Reasonable effort on the homework involves, at minimum:

- Writing code for each problem
- Passing all the doctests for some percentage of the problems (at least 50%).
- Showing that multiple attempts were made at solving the remaining problems which can be demonstrated by:

- Modifying erroring code to pass more doctests
- Working on problems for an extended period of time (i.e. there is evidence that work was done on the homework in advance of the night it is due)
- Passing a subset of the doctests (for specific problems)

You are responsible for writing solutions yourself. **If you are stuck on a problem, come get help instead of copying the answer from someone else or the Internet; you'll still get credit and won't be flagged for cheating.**

Projects

Projects are larger assignments intended to combine ideas from the course in interesting ways. Some projects can be completed in pairs. When working in pairs, you should work together to ensure that both of you understand the complete results. We recommend finding a project partner in your section. Your TA will help. You may also work alone on all projects, although partners are recommended for the paired projects. Projects are graded on both correctness and composition (composition.html).

Slip Day

You have one slip day “virtual token” used to give you more time for homework or projects. This can be used once for any homework or project to extend the deadline by 24 hours. This cannot be used to extend project checkpoints or extra credit deadlines - it can only be used to extend the final deadline. Slip days also cannot be applied to labs.

Exams

The first midterm exam will be held the evening of Thursday, July 18. You are permitted to bring two double-sided, letter-sized, handwritten sheets of notes.

We will allow alternate midterms only for conflicts with UC Berkeley exams. For students with another valid excuse for missing a midterm, notify us *before* the exam in question and we will weight your final to account for the missing exam. Note that missing the midterm will mean that the final will account for about 45% of your grade. The midterm must be taken in-person on the UC Berkeley campus.

The final exam will be held the evening of Thursday, August 15. You are permitted to bring three double-sided, letter-sized, handwritten sheets of notes.

If you have a direct conflict with another final exam, we will allow you to take an alternate final. We will not provide final alternates for any other reason. We will release a form to account for exam conflicts.

Resources

Textbook

The online textbook for the course is Composing Programs (<http://composingprograms.com/>), which was created specifically for this course. Readings for each lecture appear in the course schedule. We recommend that you complete the readings before attending lecture.

Computing Resources

If you are enrolled in the class, you may request a CS 61A instructional account. This account will allow you to use any EECS instructional lab computer in Soda or Cory Hall. You may use any lab you wish, as long as there is no class using the space.

Be respectful of the lab space. Please don't steal the chairs, and definitely do not eat or drink in the lab. Don't unplug anything; unplugged computers make our hard-working instructional computing team very sad. If you see someone disrupting the space, ask them to stop.

Labs are normally available for use at all times, but you need a card key for evening access. If you are a Berkeley student, your student ID will automatically grant you access to the Soda second floor labs. Otherwise, you can fill out an application from 387 Soda (the front desk).

Grading

Your course grade is computed using a point system with a total of 300 points.

- The midterm, worth 60 points.
- The final exam, worth 80 points.
- Homework, worth 30 points.
- Four projects, worth 110 points.
- Lab Assignments, worth 20 points.

There will be no drops for homework assignments, but of the mandatory (non-review) labs, only 10 contribute to your score.

There are a handful extra credit points available throughout the semester, perhaps around 10, that are available to everyone.

Each letter grade for the course corresponds to a range of scores:

A+	≥ 294	A	≥ 283	A-	≥ 270
B+	≥ 255	B	≥ 230	B-	≥ 215
C+	≥ 200	C	≥ 190	C-	≥ 180
D+	≥ 175	D	≥ 170	D-	≥ 165

Your final score will be rounded to the nearest integer before being converted to a letter grade. 0.5 rounds up to 1, but 0.49 rounds down to 0.

There is no curve; your grade will depend only on how well you do, and not on how well everyone else does. Score thresholds are based on how students performed in previous semesters. It is possible that the instructors will adjust the thresholds in your favor, for

example if exam scores are abnormally low, but that scenario is unlikely. More likely, these are the exact thresholds that will be used at the end of the course to assign grades (contrary to popular rumor).

Incomplete grades will be granted only for dire medical or personal emergencies that cause you to miss the final, and only if your work up to that point has been satisfactory. You must complete all coursework before the drop deadline to be considered for an incomplete grade.

Lab and Discussion Participation

Attending a discussion or lab section will earn you one participation credit. Participation credits can be used to recover points on exams. A total of 20 participation credits can be used for this purpose - that is, attending 20 discussion and lab sections will give you the maximum amount of exam recovery.

We expect there to be about 14 lab and discussion sections across the semester. Therefore, you can miss a total of 8 sections and still receive the maximum exam recovery.

We calculate your exam recovery using the following logic, where `participation` is the number of participation credits you earn (out of 28):

```
def exam_recovery(your_exam_score, participation, max_exam_score, recovery_cap=20):  
    half_score = max_exam_score / 2  
    max_recovery = max(0, (half_score - your_exam_score) / 2)  
    recovery_ratio = min(participation, recovery_cap) / recovery_cap  
    return max_recovery * recovery_ratio
```

According to this formula, if you receive more than half the available points on each exam, then you don't recover any points. If you score just below half the points, you will recover a few points. If you score far below half the points, you will recover many points. The more recovery credits you earn, the more exam points will be recovered.

Additionally, what matters for exam recovery is the percentage of the total participation credits you receive, not the absolute number of participation credits.

The purpose of this policy is to ensure that students who continue to invest time in the course throughout the semester are able to pass.

Exam



Your Exam Score

Participation Credits

Calculate

Points Recovered

Adjusted Exam Score

Late Policy

If you cannot turn in an assignment on time, contact your TA and partner as early as possible. Depending on the circumstance, we may grant extensions.

- **Labs:** We very rarely accept late lab submissions. There is no partial credit.
- **Homework:** We very rarely accept late homework submissions. There is no partial credit.
- **Projects:** Submissions within 24 hours after the deadline will receive 75% of the earned score. Submissions that are 24 hours or more after the deadline will receive 0 points. Each question is worth some points, so it is possible to earn partial credit on a project.

Again, you have one slip day "virtual token" which you can use to extend the final deadline for a homework assignment or project by 24 hours.

Learning Cooperatively

With the obvious exception of exams, we encourage you to discuss course activities with your friends and classmates as you are working on them. You will definitely learn more in this class if you work with others than if you do not. Ask questions, answer questions, and share ideas liberally.

Learning cooperatively is different from sharing answers. You shouldn't be showing your code to other students, except to your project partner or to someone who has already submitted the assignment and is helping you finish. If you are helping another student, don't just tell them the answer; they will learn very little and run into trouble on exams. Instead, try to guide them toward discovering the solution on their own. Problem solving practice is the key to progress in computer science.

Since you're working collaboratively, keep your project partner and TA informed. If some medical or personal emergency takes you away from the course for an extended period, or if you decide to drop the course for any reason, please don't just disappear silently! You should inform your project partner, so that nobody is depending on you to do something you can't finish.

Online Forum

If you have any questions, please post them to Piazza (<http://www.piazza.com/berkeley/summer2019/cs61a>), the course discussion forum. Piazza allows you to learn from questions your fellow students have asked. We encourage you to answer each others' questions!

Piazza is the best and most reliable way to contact the course staff. You are also welcome to email cs61a+su19@berkeley.edu, an instructor, or your TA directly.

Academic Honesty

Cooperation has a limit, and in CS 61A that limit is sharing code. You are free to discuss the problems with others beforehand, but you must write your own solutions. The only student with whom you can share code is your project partner.

Since this may be your first computer science class, exactly what constitutes as cheating might be unclear. If you are unsure if what you are doing is cheating, please clarify with the instructors or TAs. The following is a list of things you should NOT do. This list is not exhaustive, but covers most of the big offenses:

- Do not copy code from any student who is not your partner.
- Do not allow any student other than your partner to copy code from you.
- Do not copy solutions from online sources such as Stack Overflow, Pastebin, and public repositories on GitHub.
- Do not post your solutions publicly during or after the semester.

If you find a solution online, please submit a link to that solution anonymously (<https://goo.gl/forms/nL2yOj1Z81HcQYDi2>). When we find an online solution, we ask the author to remove it. We also record the solution and use it to check for copying. By reporting online solutions, you help keep the course fair for everyone.

In summary, we expect you to hand in your own work, take your own tests, and complete your own projects. The assignments and evaluations are structured to help you learn, which is why you are here. The course staff works hard to put together this course, and we ask in return that you respect the integrity of the course by not misrepresenting your work.

Rather than copying someone else's work, ask for help. You are not alone in this course! The entire staff is here to help you succeed. If you invest the time to learn the material and complete the projects, you won't need to copy any answers.

A Parting Thought

Grades and penalties aren't the purpose of this course. We really just want you to learn. The entire staff is very excited to be teaching CS 61A this semester and we're looking forward to meeting such a large and enthusiastic group of students. We want all of you to be successful here. Welcome to CS 61A!

CS 61A ([/~cs61a/su19/](#))

[Weekly Schedule \(/~cs61a/su19/weekly.html\)](#)

[Office Hours \(/~cs61a/su19/office-hours.html\)](#)

[Staff \(/~cs61a/su19/staff.html\)](#)

Resources ([/~cs61a/su19/resources.html](http://~cs61a/su19/resources.html))

[Studying Guide \(/~cs61a/su19/articles/studying.html\)](http://~cs61a/su19/articles/studying.html)

[Debugging Guide \(/~cs61a/su19/articles/debugging.html\)](http://~cs61a/su19/articles/debugging.html)

[Composition Guide \(/~cs61a/su19/articles/composition.html\)](http://~cs61a/su19/articles/composition.html)

Policies ([/~cs61a/su19/articles/about.html](http://~cs61a/su19/articles/about.html))

[Assignments \(/~cs61a/su19/articles/about.html#assignments\)](http://~cs61a/su19/articles/about.html#assignments)

[Exams \(/~cs61a/su19/articles/about.html#exams\)](http://~cs61a/su19/articles/about.html#exams)

[Grading \(/~cs61a/su19/articles/about.html#grading\)](http://~cs61a/su19/articles/about.html#grading)