Course Information

Course Description

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. This first course concentrates mostly 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. The next course, CS 61B, will deal with the more advanced engineering aspects of software, such as constructing and analyzing large programs. Finally, CS 61C concentrates on machines and how they carry out the programs you write.

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. Mastery of a particular programming language is a very useful side effect of studying these general techniques. However, our hope is that once you have learned the essence of programming, you will find that picking up a new programming language is but a few days' work.

Course Format

The course includes many events: lecture, lab, discussion, office hours, guerrilla sections, homework parties, project parties, and review sessions. Weekly lab and discussion sections are typically the most valuable events to attend. For the rest, your attendance is optional. These events exist to help you learn, and so you are advised to attend as many as required to master the material.

Lecture: The course includes three 50-minute lectures per week. Lectures are screencast (voice and slides). You are welcome to view the content either live or on video, but most students prefer to stay home.

Screencasts of the live lecture will be published approximately 36 hours after each event, and links will appear in bCourses.

The first two live lectures will be held in Zellerbach Auditorium. All other live lectures will be held in Pauley Ballroom (2nd floor of MLK Student Union). There are no screencasts for those lectures.

Lab and Discussion Sections: The course includes one laboratory section and one discussion section each week. These sections are run by an amazing group of undergraduate student instructors (UGSIs) who have been carefully selected for their ability, enthusiasm, and dedication to learning. Getting to know your UGSI is an excellent way to succeed in this course. UGSIs are also often called teaching assistants (TAs).

Please attend the section in which you have enrolled. You have priority to sit in that section. If you are waitlisted, please attend the section for which you are waitlisted. You are welcome to take a seat if one is available. If there aren't open seats, your UGSI might let you sit on the floor.

It is desirable that you attend matching discussion and lab sections (last two digits of the section numbers match). The same TA generally handles both of these, which means that he or she will be more familiar with you, and vice-versa.

You can also attend a section for which you are not enrolled or waitlisted, but only if there is room for all enrolled and waitlisted students and you. Given the large demand for 61A this semester, I don't expect that there will be much extra space in sections.

Most assignments in 61A are completed in pairs. We strongly encourage you to find a partner in your own section. Your TA will help you. You are allowed to work alone on any assignment, but life is better with a partner.

Office Hours: Attending office hours is another excellent way to succeed in this course. Office hours are held by TAs and the instructor each week. See the office hours schedule (http://cs61a.org/office-hours.html) for times and locations.

In office hours, 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.

Optional Sections: In addition to the weekly class meetings, the course includes many optional events that are designed to help you master the course material and complete the assignments. Details of these events will be announced as they approach.

Enrollment

In order to enroll in 61A, you must enroll in both lecture and a section. There is room in lecture for everyone who finds a seat in a section. If you're at the end of a long section waitlist, that's bad. You should consider switching to a section that has fewer students attempting to enroll. My hope is that everyone who wants to take this course will be able to enroll.

Concurrent enrollment applications will be considered once the regular waitlist is empty.

For students who are not enrolled but want to join the course (on the waitlist, concurrent enrollment, or auditing), please turn in all assignments by their due date. Late work will not be accepted.

If you cannot enroll because of a scheduling conflict, you will need to talk to a campus advisor.

Extra Units

There are two different 1-unit courses that are designed to be taken concurrently with 61A. You can enroll in either one or both, but most students don't enroll in either.

CS 97: Additional Discussion of the Structure and Interpretation of Computer Programs is a unit offered to students who agree to attend weekly mentoring sessions. These small group sessions include 3 to 5 students and a course tutor. These focused sessions are an excellent way to keep up with and master the course material.

CS 98: Additional Topics on the Structure and Interpretation of Computer Programs is a weekly 1-unit lecture series on nifty optional content. Attendance is required. There are a few homework assignments. This course is recommended for students seeking an extra challenge. Students who aren't enrolled are also welcome to show up.

Enrollment and scheduling information will be announced once the main course begins.

Materials

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 calendar (http://cs61a.org/). We recommend that you complete the readings before watching the lecture videos or attending live lecture. Watching videos first, and then reading the book works better for some students. Ignoring the book entirely is a bad idea.

In addition, the course website links to all guides, handouts, and practice materials available for the course.

Programming Language

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.

In addition, you will learn a subset of the Scheme and SQL languages.

Prerequisites

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

There is no formal programming-related prerequisites for admission to 61A, but that doesn't mean that it's the right first course for all students. Many 61A students have had significant prior programming experience, but many do not. Students without prior experience typically spend **a huge amount of time each week** on the course.

If you don't feel ready for 61A, we recommend that you take one of these courses first. You can always take 61A in a future semester; it's even offered over the summer.

- CS 10: The Beauty and Joy of Computing (http://cs10.org/) is an introduction to computer science for non-majors (and majors needing more programming experience). The course teaches students how to program using Snap (based on Scratch), one of the friendliest programming languages ever invented. It's purely graphical, which means programming involves simply dragging blocks around, and building bigger blocks out of smaller blocks. But the course is far more than just learning to program! You'll learn some of the "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: The Foundations of Data Science (http://data8.org/) is an introduction to data science designed to be accessible for all Berkeley students. 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, but the programming skill you will acquire in Data 8 will prepare you for the fast pace of CS 61A.
- Udacity CS 101: Intro to Computer Science (https://www.udacity.com/course/intro-to-computer-science--cs101) is a free online course that introduces computer science using the Python 2 programming language. The course provides solid explanations of how programming languages work along with lots of exercises. Working through lessons 1 through 3 before starting 61A is a great way to give yourself a head start in the course.

If you have substantial prior programming background, you may feel that you can skip 61A. In most cases we don't recommend that, but meet with your instructor in office hours to discuss the issue. Although 61A is the first course in the CS sequence, it's quite different from most introductory courses. Even students with years of prior experience are typically not bored in 61A. Prior experience does allow some students to skip 61B, which is more comparable to courses taught elsewhere.

Ask Questions!

Your first and most important resource for help in learning the material in this course is your fellow students. Work closely with your project partner. You are responsible for helping each other learn.

If you have questions that others might have as well, regarding projects, homeworks, course policies, etc., post your questions to Piazza (https://piazza.com/berkeley/spring2017/cs61a), the course discussion forum. Piazza allows you to answer questions from other students.

You should also ask many questions of the course staff, especially during lab sections and office hours. In addition to instructor and TAs, the course includes a staff of tutors and Lab Assistants (LAs). Each LA will have scheduled hours to be in the lab. Tutors are available periodically to assist you with your assignments at homework and project parties. They also host guerrilla sections, which go over past material by topic.

The best way to learn in this course is to attempt all assignments on your own, but ask questions frequently before you get stuck or frustrated.

You are welcome to email your instructor or TA directly. Contact information appears on the staff page of the course website.

Computer Resources

The computing laboratories in 271, 273, 275, 277, and 330 Soda are our primary lab rooms, although the CS 61A accounts can also be used from any EECS Instructional lab in Soda or Cory Hall. The lab is normally available for use at all times, but you need a card key for evening access to the lab.

Current UCB students: If you are enrolled in the course, your Cal Student ID serves as your card key and will automatically be activated for access to the Soda second and third floor labs (including entering the building). You do not have to do anything, unless for some reason it doesn't work, then see below.

Concurrent/other students: You can fill out an application and obtain a white card key from 387 Soda Hall (the front desk). There is a small fee for access.

During scheduled lab sessions, only students enrolled in that particular section may be in the lab. At other hours, any 61A student may use the lab on a drop-in basis.

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.

Projects and Homework Assignments

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

- **Laboratory exercises** are short, relatively simple exercises designed to introduce a new topic. You can complete these during the scheduled lab meetings and submit them then.
- **Homework assignments** have various kinds of questions. Regular questions are more involved than lab and are meant to illustrate and explore new topics. Vitamins are short problems that verify you have watched the previous week's lecture. Quiz questions are a way to assess your progress in the course. Both Vitamins and Quiz questions should be completed alone. You are encouraged to discuss regular homework questions with other students, but your final solution should be submitted alone.
- **Projects** are larger assignments intended to teach you how to combine ideas from the course in interesting ways. There are four projects during the semester. You are encouraged to complete projects in pairs; your partner should be another student in your section.

The purpose of the homework is for you to learn the course material, not to prove that you already know it. You get full credit for getting some threshold fraction of the required problems right.

Each homework is worth two points for a reasonable effort, zero points for a missing homework or one that seems to show no effort, or negative ten (-10) points for a solution copied from someone else. Detecting copied homework is very easy for our computers, so don't try.

The four programming projects are graded on the correctness and clarity of your solutions. If you work with a partner (which you should), work together to ensure that both group members understand the complete program you create.

Tests and Grading

Your course grade is computed using a point system with a total of 300 points, with the following distribution.

- Midterm 1, worth 40 points.
- Midterm 2, worth 50 points.
- The final exam, worth 80 points.
- Four projects, worth a total of 100 points.
- Homework, quizzes, labs worth a total of 30 points.

Midterms are held in the evenings and are two hours long. They are open notes, but calculators and computers are not allowed.

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

```
A+  289+       A  269-288     A-  255-268
B+  235-254    B  215-234     B-  200-214
C+  190-199    C  175-189     C-  170-174
D+  165-169    D  155-164     F     0-148
```

This grading formula implies that *there is no curve*; your grade will depend only on how well you do, and not on how well everyone else does. In a typical semester, about 1/3 of students receive A's and another 1/2 of students receive B's, but these ratios are not fixed.

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.

Discussion Participation

Attending and participating in discussion section may allow you to earn back lost points on your midterms. Therefore, participation is tracked and potentially beneficial. Use the calculator below to determine how many points you can earn back. The discussions that count towards each exam are:

- **Midterm 1 Participation:** Discussions 1-4 (excluding discussion 0)
- **Midterm 2 Participation:** Discussion 5-7
- **Final Participation:** TBA

Exam:

Midterm 1

Your Exam Score:

0

Discussions:

0

Calculate

Points Recovered:

Adjusted Exam Score:

We calculate your recovery points using the following logic:

```
def recovery_points(your_score, discussions, max_exam_score, max_discussions):
    if discussions == 0:
        return 0
    else:
        half_score = max_exam_score / 2
        max_recovery = (half_score - your_score) / 2
        return max(0, max_recovery * discussions / max_discussions)
```

Learning Cooperatively

With the obvious exception of exams and take-home quizzes, we encourage you to discuss *all* of the 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.

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.

Academic Honesty

Cooperation has a limit, however, and in 61A that limit is sharing code. Feel free to discuss the problems with others beforehand, but not the code that solves them. Homework and projects can be completed in pairs. You can share everything with your partner. Do not share your code with anyone but your partner, and do not read anyone but your partner's code. Do not post your solutions online. Do not use pastebin or github, which post your work publicly by default. Do not read solutions that you find online. Write your own programs and keep them to yourself.

If you find a solution online, please submit a link to that solution (http://goo.gl/GEHk49).

I 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're 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.

The EECS Department Policy on Academic Dishonesty says, "Copying all or part of another person's work, or using reference materials not specifically allowed, are forms of cheating and will not be tolerated." The policy statement goes on to explain the penalties for cheating, which range from a zero grade for the test up to dismissal from the University, for a second offense.

Rather than copying someone else's work, ask for help. You are not alone in this course! The TAs, lab assistants, and instructor are all 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.

Early Policy

On each project, you can earn an additional bonus point by submitting the project at least 24 hours before the deadline.

Late Policy

If you cannot turn in an assignment on time, contact your TA and partner as early as possible. Late project submission requires approval by your TA. Typically, projects will be accepted up to 24 hours after the deadline, but late projects will only receive 2/3 of the earned score. No credit will be given for late homework or for projects turned in more than one day late. Exceptions may be made for *extraordinary* circumstances. Otherwise, *regardless* of how incomplete your assignment may be at the time, submit whatever you have before the deadline or, if absolutely necessary, within 24 hours after that.

A Parting Thought

This document shouldn't end with a list of late penalties, because penalties and grades aren't the purpose of the course. We actually just want you to learn. We're very excited to have such a large and enthusiastic group of students this semester. We want all of you to be successful here. Welcome to 61A.