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, 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. Lecture screencasts are distributed online after each live lecture. You are welcome to view the content either live or on video.

Links to lecture screencasts will appear on CalCentral approximately 36 hours after each event.

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 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.

For the first week, please go to the discussion & lab you are enrolled in. You If you wish to switch sections later, there will be a form that you can fill out to request a switch. You do not need to update your section on Bear Facts. Seating priority will be given to people who are officially

enrolled in a section, but we will all do our best to accommodate your wishes.

It's best to find a permanent section as soon as you can and in the first week. Course projects are completed in pairs. We strongly encourage you to find a partner in your own section. Your TA will help you.

Participation in lab and discussion is tracked. See the section on grading for more details.

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. A schedule appears on the staff page of the course website.

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.

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 schedule. We recommend that you complete the readings before watching the lecture videos or attending live lecture.

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

Programming Language

61A 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.

Prerequisites

Math 1A is a corequisite for 61A. (That is, it may be taken concurrently.)

There is no formal programming-related prerequisites for admission to 61A. Many 61A students have had significant prior programming experience, but many do not. However, students without prior experience typically spend a large amount of time each week on the course. There is no need for you to be familiar with any particular programming language.

If you don't feel ready for 61A, we recommend that you take CS 10: The Beauty and Joy of Computing (http://inst.eecs.berkeley.edu/~cs10/), which is an introduction to computer science for non-majors (and majors needing more programming experience). The course will teach 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.

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 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 or 61C, which are more comparable to courses taught elsewhere.

Computer Accounts

To set up an account on the lab computers, you need an account form, which you will receive in the first lab.

A lab account can be useful for this class if you do not plan to use your personal computer for the course assignments.

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 (http://www.piazza.com/class), 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 projects. 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, and 277 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 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 Homeworks

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** are more involved and are meant to illustrate and explore new topics. Homeworks are generally assigned on Wednesdays and due the following Wednesday at 11:59pm. You are encouraged to discuss the homework with other students, but your final solution should be developed alone, unless otherwise indicated.
- **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. (Working alone or with someone from another section is allowed, but not a good idea.)

The purpose of the homework is for you to learn the course material, not to prove that you already know it. Therefore, the weekly homeworks are not graded on the correctness of your solutions, but on effort. You will get full credit for an entirely wrong answer that shows reasonable effort!

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 homeworks 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.

- Two midterms, worth 40 and 50 points.
- The final exam, worth 80 points.
- Four projects, worth a total of 95 points.
- Homework, quizzes, labs worth 35 points.

Midterms are held in the evenings and are two hours long to give you more time to complete them than lecture would allow. Exams are open book open note.

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     D–  150–154
```

This scale is nonlinear; the steps are wider in the B range. In a typical semester, about 75% of students receive A's and B's.

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.

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.

Lab/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. Details of the policy will be released closer to the midterms.

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. 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.

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.

Early Policy

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

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.