CS106A Syllabus

CS106A introduces code and computer programming for people who have not programmed before. Code and programming are central to so much in modern life, yet code can appear to be impossibly opaque. By working gradually and with its army of section leaders, CS106A takes students into the world code, building things they could not have imagined.

Staff

Instructor: Nick Parlante, nick.parlante@cs.stanford.edu

Head TA: Elyse Cornwall cornwall@stanford.edu
Contact Elyse for all sorts of course help.

See the course page for office hours, and as a first step, Nick and Elyse are always available right after lecture - feel free to bring your laptop down and chat.

Then we have many section leaders to lead the small sections and provide lots of helper hours for when students get stuck (see below).

Topics

We will cover all the important topics of basic programming in Python: types, numbers, strings, functions, linear collections, dictionaries, logic, decomposition, good programming style, whole-program structure, text, file-processing, debugging, and performance. We'll also touch on more advanced topics you might want in the future, including lambdas, comprehensions, modules, and matplotlib.

Python is a huge language with many advanced features, and CS106A does not cover all of Python's features. CS106A teaches the important core features, and you will be able to solve real programming problems with just this course.

Other Courses:
CS105 CS106A/B CS193Q CME193 CS106AX

CS106A is the main, first course in programming and computer science, for people who with zero experience.

CS106B is the second course, teaching more advanced programming and computer science for people who know basic programming.

CS105 is a more lightweight introduction to CS ideas, but without as much coding as CS106A.

CME193 is a course in applied Python for scientists and engineers. It is a course one could take after CS106A.

Ways and Units

CS106A satisfies the university WAYS-FR requirement if taken for a letter grade. Undergraduates should sign up for 5 units. Graduate students have the option of taking it for 3-5 units, but the workload and grading is unchanged. The 3-unit option is just a courtesy for grad students who have a cap on the number of units they can take.

Section - Sign Up Thu

There will be a weekly section to study and practice code in a small group setting with individual help. Participating in section is 5% of the course grade. Signups begin Thu of week-1, see the course page for details and the link to sign up. You will have your own section leader for the quarter who will lead section and grade your homeworks.

Wed section note: the section problems will often use the lecture material from Wed. Therefore, only sign up for a Wed section if you tend to be keeping up with live lecture.

Section Leaders and the Lair

CS106A has a large staff of highly capable section leaders. The large staff enables us to staff the small weekly sections, and also allows CS106A to have **a lot** of helper hours at the so called "Lair" - see the "Getting Help" section on our course page.

Python Resources

In lecture we'll have links to some online code exercises on Nick's experimental server. Other times we will distribute a .zip file with that day's code examples.

We'll use web resources as our Python reference as we go, and Nick maintains a free Python Guide with more detail on Python topics (linked off the course page too).

There is no required textbook, and a book is not needed for this course. However, if a student who want a recommendation, "Introducing Python" by Lubanovic is a

good introductory book.

Python 3 / PyCharm

We will use Python version 3. We will have instructions to install it in week 2. At the start we'll use the experimental server which works without installing anything. Later we will do larger exercises where you will need a computer with Python 3 installed on it. You will also need to install the free PyCharm development environment. We'll have detailed instructions for that when we get there.

You need to have a computer (not a chromebook or ipad) to do the work for this class. Stanford can loan you one if needed. See Stanford Tech Lathrop

Lab In Lecture

We will experiment integrating little exercises within lecture, so expect to do some exercises live as we go. Education research shows that doing a little activity with what you just saw helps a lot with learning. (see Carl Wieman, Stanford school of ed.)

Homework

We will have weekly programming assignments. Usually homework will go out by Thu and be due the following Tue or Wed night at 11:55pm.

Exams

We'll have a midterm and a final. The exam problems will be very similar to the coding problems on the homeworks, so you can think of the homeworks as a first practice area, getting the concepts down to get ready for the exams. We'll provide lots of practice problems before the exams.

Grading

The course grade is based on homeworks and exams. We curve the exam scores.

Homework 40%

Section 5%

Exams 55%

Honor Code

In the spirit of collegial and cooperative learning, you are free to discuss ideas and approaches with other students, and then write your own solution code. The key is this: **all the code you submit you should type in and get working yourself**. In particular, it is not ok to share or paste in someone else's code or get code from a previous quarter. You should not be looking at another student's homework code.

For discussion or tying out ideas, the many lecture examples work well and of course it's fine for everyone to look at and experiment with that code.

Web search: it's fine if you search the web to find the right 2 line phrase to solve something, like "sort strings" - programmers do that sort of search all the time, and finding and using short phrases like that is fine. Do not, however, search for a whole homework function and paste in what you find. We want you to write that code.

The Computer Science department produces many honor code cases at Stanford. This is not because CS is a magnet for cheating; it's just that online submissions provide a large body of evidence, and computer science has tools which do an extremely good job of finding cheating.

If you think a bit of collaboration may have crossed the line, please mention the collaboration at the top of your homework file so your section leader knows. This will most likely not change anything about your grade, you're just getting it on the record. You can never get in honor code trouble for collaboration clearly described in this way.

As mentioned above, CS106A exams will very much resemble the homework problems. So that's an additional reason it's best to author and understand your own code.

On a related note, when you are done with a homework - please don't post your code on the internet! That causes problems for us and for people trying to learn the material later.

In effect we have a big safe-harbor of "talk about ideas, write your own code". It turns out, students who discuss things and then write their own code produce solutions that may be similar but with many little differences too. The position of CS106A is that this is fine.

However, sometimes a student will paste in a copy of the code from somewhere, and then eventually we call them on it, perhaps a little after the quarter is over. Then the student looks at the honor code statement, and makes up this story - "well the code is the same for all those lines because we talked about each line, word by word, and talking is allowed." So we must add — if the discussion is so detailed,

word by word, that we get big sections of identical code, that is not within the safe harbor. That is just copying.

Philosophy and the Honor Code

It's not that people can be divided into cheaters and non-cheaters in some pre-ordained way, though that is an easy way to think about life. It's more that the stress and bad decision making of a particular *situation* make a cheater of someone. If you feel you are in that position, contact Nick or the TA and we promise we'll work something out where you can pass this class vs. making a huge mistake.

Lateness

1. Worked turned in by the due date will get a 2% on-time bonus. The 2% is small enough that will not make much difference to anyone's grade, but it's a reward for people who start the work early enough to finish on time.

2. After the due date, each assignment will have a full-credit "grace period", 48 hours unless documented otherwise on the assignment. You do not need any permission to use the grace period. Just turn the work in. We have this because we want students to have a little extra time when needed to get a good solution working.

3. Work turned in later than 48 hours can get some credit, but loses about 15% per day beyond the grace period. If you turn something in very late, please contact your section leader so they know it's there to grade. With the 15% penalty going, it's probably better to turn in what you could get working for some credit and move on to the next assignment. We have a lot of assignments. And of course contact the head TA to arrange extra time for extraordinary circumstances. Please email the TA before classtime if you want a response that day. Work does not need to be perfect or complete to get some credit, so turning in something partial is a workable strategy.

Topics and Weekly Schedule

Below is an approximate topic plan. There is plenty of time in 10 weeks to cover all the important Python topics we need. The exam dates are fixed and you need to be able to attend those.

- Week-#/Monday-Date
- Week 1, Mon=Jan 9 - Class begins, Bit problems to start: code, syntax errors, logic, loops, functions, decomposition, style 1.0

- Week 2, Jan 16 - (MLK Holiday Mon) RGB colors, digital images, expressions, image processing, loops, image-logic, bluescreen
- Week 3, Jan 23 - black-box functions, strings, Doctests, 2D Grid
- Week 4, Jan 30 - string find/slice, files, files processing, print, standard output, start lists
- Week 5, Feb 6 - More complex loops, nested loops, computer hardware
- Week 6, Feb 13 - **Midterm Mon Feb 13, 7pm** more algorithms, drawing, double while
- Week 7, Feb 20 - (Presidents Holiday Mon) dict count algorithm, nested dict algorithms
- Week 8, Feb 27 - Mapping, lambda, sorting, custom sorting, modules 1.0
- Week 9, Mar 6- Modules 2.0, matplotlib, Advanced topics
- Week 10, Mar 13 - Last week, Conclusions, Life after CS106A
- **Final Exam Mon Mar 20 8:00-11:00am** (our exam will actually be less than 3 hours) Note that CS106A has a special exam time slot, at the very start of exams