Lecture 1: Introduction

Marvin Zhang
06/20/2016

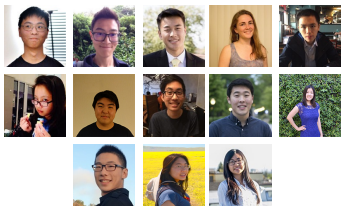
Welcome to Berkeley Computer Science!

Humans of CS 61A

2 Lecturers

12 TAs

13 Tutors

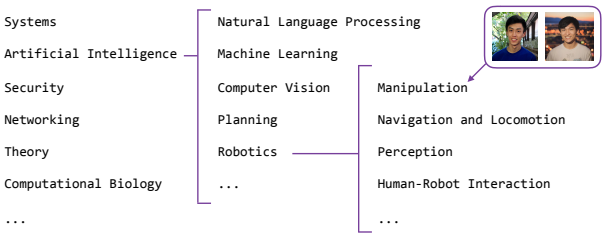


100+ Lab assistants!

400+ Students!!!

Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?



CS 61A in one slide

- High-level ideas in computer science:
  - *Abstraction*: manage complexity by hiding the details
  - *Paradigms*: utilize different approaches to programming

- Master these ideas through implementation:
  - Learn the Python programming language (& others)
  - Complete large programming assignments
- A challenging course that will demand a lot from you

Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing
cs10.org
Offered this summer!

Data Science 8: Foundations of Data Science
data8.org

Course Policies

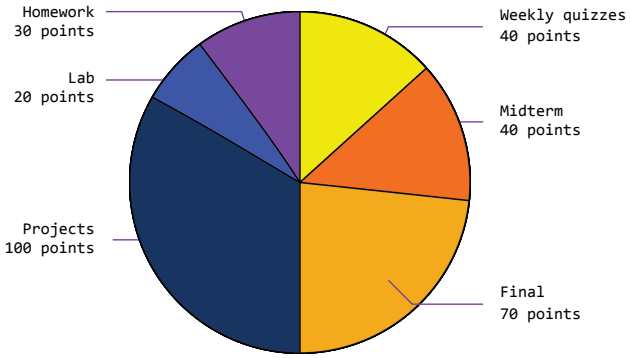
Details on cs61a.org

Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com

- Regular homework assignments
- 4 big programming projects
- Weekly quizzes, one midterm, and one final exam
- Lots of special events!

Grading



A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys
- 12 (graded) lab assignments, 2 points each
  - Two lowest lab scores will be dropped
- Written quizzes will be *in lecture* on Thursdays
  - We have sent out instructions for students who cannot attend Thursday lectures
  - One written or coding quiz score will be dropped
- This class is *not* curved!
  - *Collaboration*, not competition

The limits of collaboration

- Everyone should give and receive help, because everyone benefits and learns
- There is only one rule:
  - *Your code is yours, and yours only.*
- This means that:
  - You *cannot* copy or use code from anyone except your partner
  - You *cannot* share your code with anyone except your partner
- Share and discuss *ideas*, not code
- Build good habits now!

Getting help

- Discuss everything in the course, except exams, with your partner and your classmates
  - *Teaching* is the best way to learn
- Ask and answer questions on Piazza
- Use the course staff! We're here to help you learn
  - Labs and office hours are the perfect time to talk to the lecturers, TAs, tutors, and lab assistants
  - Lab assistants will also be available for *checkoffs* during labs

A few last thoughts

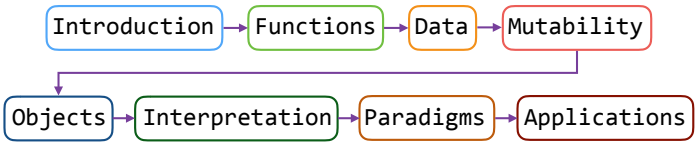
- Find all the course details and news on cs61a.org
- The most important course policy is *not*:
  - Grading
    - 75% of students in this course receive As and Bs
    - There is no curve! All of you can get an A+
  - Cheating
    - There is a community of staff and students that want you to succeed, and will help you succeed
- The most important course policy is *learning*
- Learn a lot, have fun, and welcome to 61A!

An Introduction to Programming

And, conveniently, an introduction to Python

Course organization

- Every week will center around a theme, and have a specific set of goals.



- This week (Introduction), the goals are:
  - To learn the fundamentals of programming
  - To become comfortable with Python

What's in a program? (demo)

- Programs work by manipulating *values*
- *Expressions* in programs evaluate to values
  - *Primitive expressions* evaluate directly to values with minimal work needed
- *Operators* combine primitives expressions into more complex expressions
- The Python interpreter evaluates expressions and displays their values

Mathematical expressions (demo)

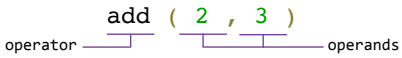
$$\lim_{x\to\infty} \frac{1}{x} \qquad sgn(x) \qquad \sin x$$

$$\sum_{i=1}^{n} i \qquad \sqrt{x} \qquad x^y \qquad \ln x$$

$$\frac{x}{y} \qquad |x| \qquad x + y$$

$$\binom{x}{y} \qquad x \mod y$$

Call expressions



- In a call expression, the operator and operands themselves are expressions
- To evaluate this call expression:
  1. *Evaluate* the operator to get a function
  2. *Evaluate* the operands to get its values
  3. *Apply* the function to the values of the operands to get the final value

Nested call expressions

```
add(add(2, mul(4, 6)), mul(3, 5))
```

- What does this call expression evaluate to?
- What are the steps that the Python interpreter goes through to evaluate this expression?

The Power of Python

Shakespeare demo!