

# Lecture 1: Introduction

June 22nd, 2020

A photograph of a modern building with a facade made of small, square tiles in various shades of gray and blue, creating a checkered pattern. Large, arched windows are visible on the upper floors. In the foreground, there are palm trees and a wooden pergola structure. The sky is clear and blue.

Welcome to CS 61A!

# CS61A

- 720 students
- 16 timezones



Humans of CS 61A



# Instructors

**Ryan Moughan**

rmoughan@berkeley.edu



**Chae Park**

chae@berkeley.edu



**Kavi Gupta**

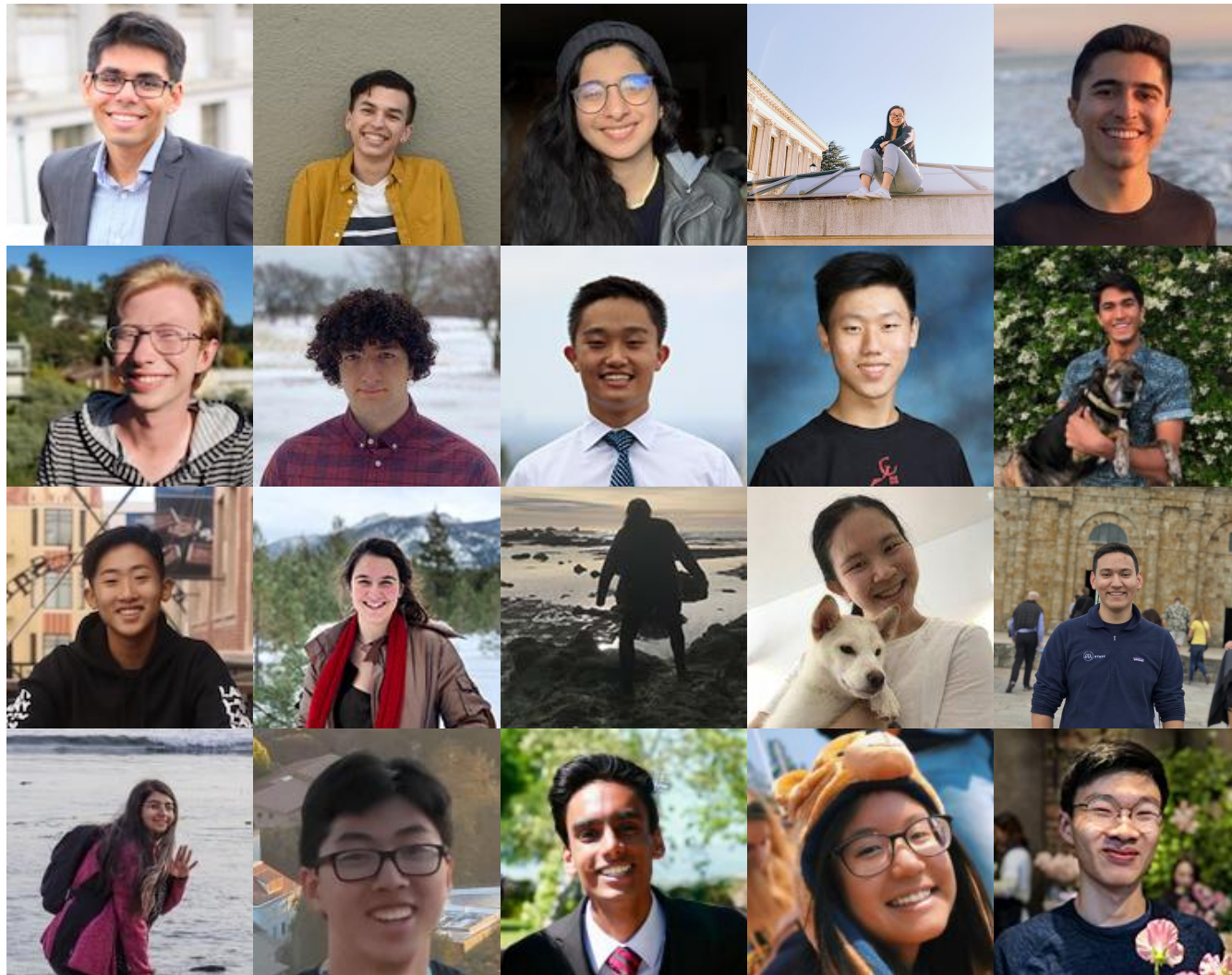
kavi@berkeley.edu



# Prof. John DeNero



# Teaching Assistants





# Tutors





You!

# Computer Science

# What is Computer Science?

- What problems can be solved using computation?
- How do we solve those problems using computers?
- What techniques lead to effective solutions?

Systems

Artificial Intelligence

Graphics

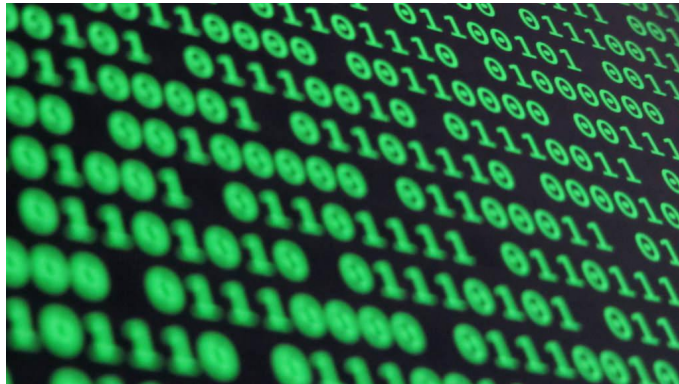
Security

Networking

Programming Languages

Theory

...



# What is CS 61A?

- A course about managing complexity
  - Mastering abstraction
  - Programming paradigms
- An introduction to programming
  - Full understanding of Python fundamentals
  - Combining multiple ideas in large projects
  - How computers interpret programming languages
- A challenging course that will demand a lot of you

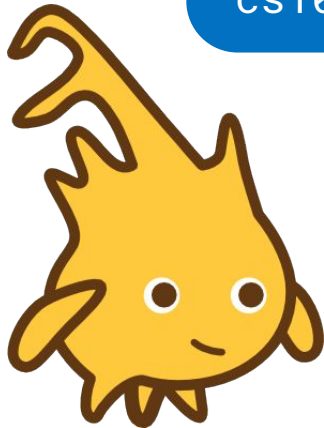


# Alternatives to CS 61A

## **CS 10 The Beauty and Joy of Computing**

*An introduction to fundamentals (& Python) that sets students up for success in CS 61A*

[cs10.org](https://cs10.org)



## **Data 8 The Foundations of Data Science**

*Fundamentals of computing, statistical inference, & machine learning applied to real-world data sets*

[data8.org](https://data8.org)

# Course Logistics

# Course Format

<b>Lecture</b>	Whenever you want!
<b>Lab</b>	the most important part of this course
<b>Discussion</b>	the most important part of this course
<b>Office hours</b>	the most important part of this course
<b>Tutoring</b>	the most important part of this course
<b>Textbook</b>	<a href="https://composingprograms.com">composingprograms.com</a>

- 8 programming **homeworks**
- 4 programming **projects**
- 1 **diagnostic quiz**, 1 **midterm exam**, and 1 **final exam**
- Lots of course support and a great community

# The First Week

- Lab 0 released today!
- Discussion starts tomorrow!
- OH starts later this week



# Lecture

- Main lectures are recorded videos by John Denero
- We will be giving supplementary live lectures once a week (details announced later)

# Discussion Section

- Only part of the course that tracks attendance
- 90 minute section twice a week
- Largely worksheet based (but do not expect to finish it)
- Recorded

# Office Hours

- Three formats: Appointments, Parties, and Instructor
- Appointment-based Office Hours:
  - 20 minutes each
  - Sign up the night before
  - Some will only be conceptual
- Parties:
  - 3 hours each
  - 2 flavors: Homework and Project
  - Queue-based
- Instructor:
  - Strong focus on concepts and not assignments

# Small Group Tutoring Sections

Small-group sections (4-5 students) centered around a worksheet which reviews content from the corresponding discussion

## **Recurring**

- Meet twice a week regularly with the same group of students
- Sign-ups will open later this week
- Start next week

## **Drop In**

- Can sign up if you feel like you could use some extra reinforcement on topics presented from the last discussion
- Sign-ups will open every end of the week



# Tools

## **Zoom: A platform for video calls**

- Can ask questions via voice or text-chat
- Option to ask questions individually in a “breakout” section
- Where discussions , hw/project parties will happen

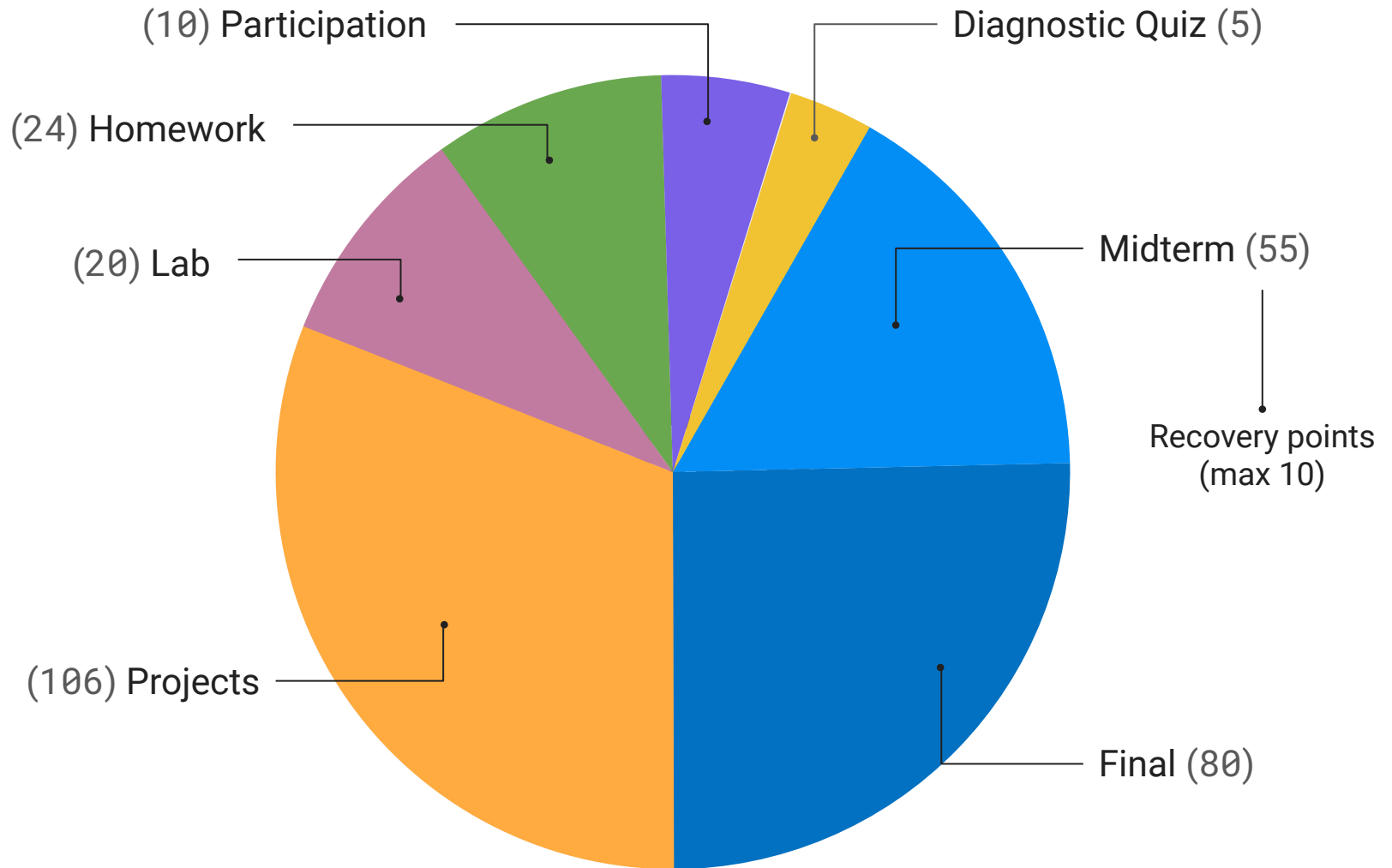
## **Piazza**

- Forum for students to post questions & get announcements from instructors

## **Various pieces of software**

- Will introduce most of these in lab00, so complete this ASAP!

# Grading



# Assignments

You can earn **150 points** through assignments:

- (10) 2 pt. Twice a week lab assignments
- (8) 3 pt. weekly programming homework assignments
- (4) 20-30 pt. programming projects

Most assignments are submitted using Ok ([okpy.org](https://okpy.org)).

You must have an [@berkeley.edu](mailto:@berkeley.edu) address listed as your primary email on CalCentral to be enrolled on Ok.

If you have not been added to OK--which you'll find out when you do your assignments--fill out the form on Piazza / email your TA ASAP

# Lab Assignments

- Generally released Mon/Wed, due Wed/Fri respectively
- Graded on correct completion, all or nothing
- Lowest two lab grades dropped
- You will complete these on your own time - however, you may find that it is helpful to work on labs during office hours to get support
- More details: <https://cs61a.org/articles/about.html#labs>

# Homework Assignments

- Generally released on Wed, due next Tues
- Graded on correctness, partial credit with every incorrect answer losing you one point on the homework (up till 0)
- **Homework Recovery:**
  - Can recovery one incorrect question per homework by going through one of hw recovery processes
    - Homework recovery session
    - Appointment based office hour
- More details: <https://cs61a.org/articles/about.html#homework>



# Participation

## Discussion Participation

- This is part of 300 points.
- You can earn up to **10 participation points**.

## Class Participation

- This is **not** part of 300 points.
- Can be used for exam recovery
- Each of the following opportunities is worth 1 class participation credit:
  - Weekly student survey (~8 possible)
  - Extra discussion section attendance (after the initial 10, 2 possible)
  - Extra lab assignment submission (after the initial 10, 2 possible)

# EPA (Efforts, Participation, Altruism)

- Extra credit(s); not part of 300 points.
- Can be earned through (but not limited to):
  - Effort = {Office hours, doing every single lab, hw, reading Piazza pages, etc.}
  - Participation = {Raising hand in discussion, asking Piazza questions, etc.}
  - Altruism = {Helping other students, answering Piazza or Office Hour questions etc.}
- Scoring will remain confidential.

# Exams

## ***Diagnostic Quiz*** (5 pts)

**When:** Thursday, July 2 @ TBA

**Format:** 60-90 minute electronic exam

## ***Midterm*** (55 pts)

**When:** Thursday, July 16 @ TBA

**Format:** 180 minute electronic exam

## ***Final exam*** (80 pts)

**When:** Thursday, August 13 @ TBA

**Format:** 180 minute electronic exam

## ***Exam recovery***

After getting >5 of class participation points, 10 credits can be used toward exam recovery points for the Midterm

## ***Alternates***

We will have alternates for each exam at a 12h offset. Fill out

<http://links.cs61a.org/alt>

# Collaboration

- This course is not curved -- collaboration, not competition, is key
- Asking questions and discussing ideas is highly encouraged
- **The only students with whom you can share code are**
  - **Your project partner**
  - **Students who have finished the problem you are working on**
- More info: <https://cs61a.org/articles/about.html#academic-honesty>

# Course Overview

Every week will center around a theme with a specific set of goals.

- Learn the fundamentals of programming
- Become comfortable with Python



Introduction

Functions

Data

Languages

Objects

Evaluation

Paradigms

Applications

Expressions

# What's in a program?

- 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

$$20 + 17$$

$$2^{100}$$

$$\sin \pi$$

$$\lim_{x \rightarrow \infty} \frac{1}{x}$$

$$f(x)$$

$$\frac{20}{17}$$

$$\sum_{i=1}^n i$$

$$\sqrt{2017}$$

$$\log x$$

$$\binom{n}{x}$$

$$|-2017|$$

An **expression** describes a computation and evaluates to a value.



# Call Expressions

Evaluation procedure for **call expressions**

1. Evaluate the **operator**
2. Evaluate the **operands** from left to right
3. **Apply** the operator (a **function**) to the evaluated operands (**arguments**)



Operators and operands are also expressions

So they also *evaluate to values*

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

# Nested Call Expressions

- Humans evaluate inside-out

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

add(add(6, 24), mul(3, 5))

add(add(6, 24), mul(3, 5))

add(30, mul(3, 5))

add(30, mul(3, 5))

add(30, 15)

add(30, 15)

45

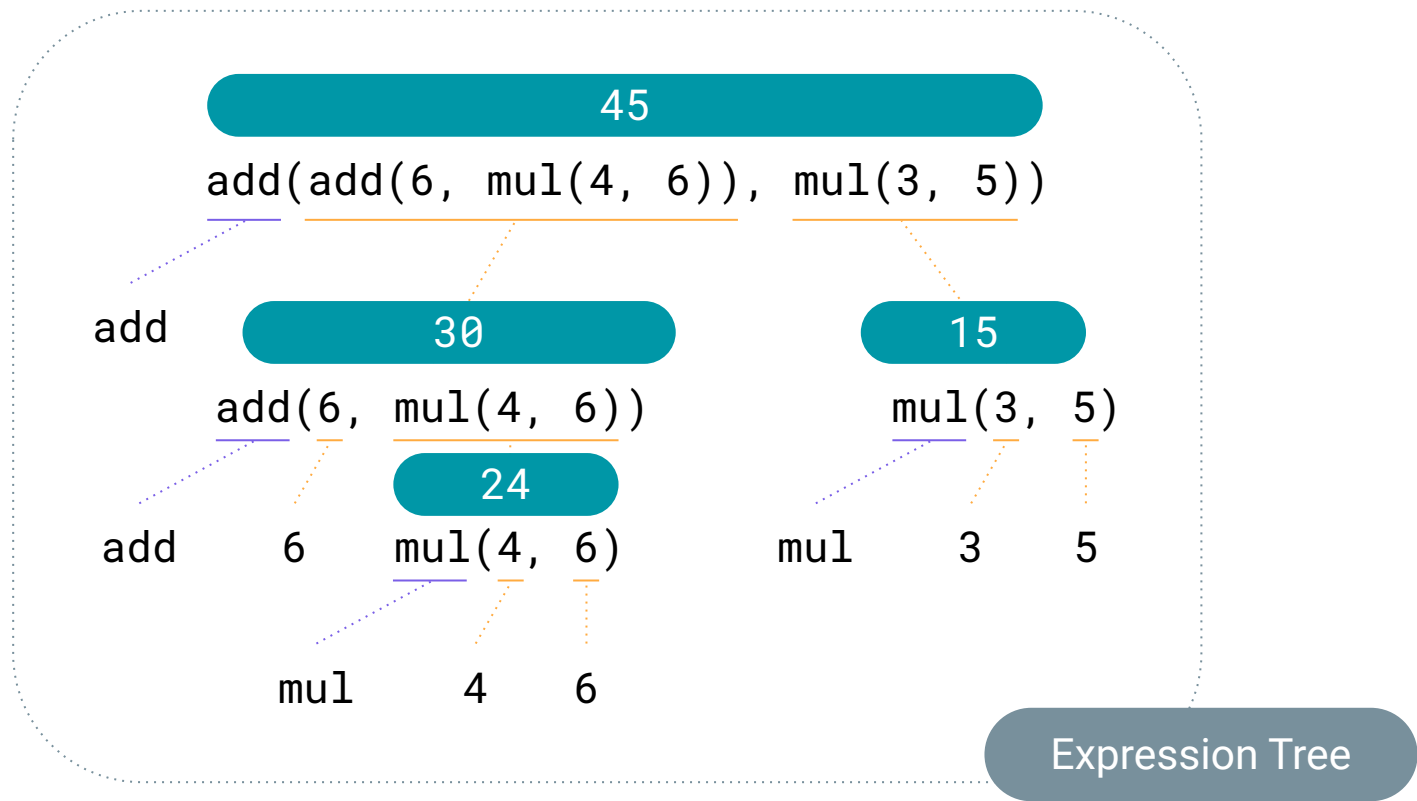
- We can jump ahead or skip around, but Python can't do that!
- How does the computer know which call to evaluate first?

# Nested Call Expressions

1 Evaluate operator

2 Evaluate operands

3 Apply!



# Functions, Objects, & Interpreters