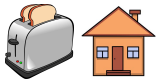
Welcome to CS 61A!

Instructors

Kay Ousterhout
(oh-stir-how-t; rhymes with )
kayo@berkeley.edu

Teaching Professor in EECS

I was a Cal PhD student (2011–2017) and built large-scale distributed systems in industry (2017–2024)

Regular office hours:

- Wednesday 1pm–2pm, Wheeler 210

John DeNero
denero@berkeley.edu

Teaching Professor in EECS

Before Cal (2014+), I was a Cal PhD student (2005–2010) and researcher @ Google (2010–2014)

Regular office hours:

- Friday 1pm–2pm, Wheeler 210

Extra instructor office hours next week! Schedule coming soon...

https://cs61a.org/fa26/staff/

About the Course

What is This Course About?

A course about managing complexity:

- Mastering abstraction
- Solving problems
- Techniques for organizing complex programs

An introduction to programming:

- Programming language fundamentals (in Python)
- Large projects to demonstrate how to manage complexity
- How computers interpret programming languages

Different types of languages: Python, Gleam, & SQL

What about Artificial Intelligence (and what's new)?

AI has changed the way programs are created.

Understanding programs and programming yourself is **extremely** useful.

The path to understanding: solve lots of problems and write lots of programs.

Using AI too much will impede this path!

We will use AI in two ways:

1. During the first 11 weeks, you write programs on your own!
   A course AI tool can help if you get stuck.

2. At the end of the semester, we'll talk about AI-assisted programming.
   (Demo at the end of lecture)

cs61a.org

How to Succeed

Lecture, Videos, and the Textbook

Videos posted to cs61a.org are essential viewing **before** coming to lecture. All of the course content will be covered in the videos.

    Get the most out of the videos by typing examples from the videos yourself!

The textbook, composingprograms.com, is written to be concise and useful. Its content is very similar to the videos.

Lecture Mon, Wed, & Fri will cover *the most important content* from the videos (but not all of it), work through examples, discuss problem-solving strategies, and give perspective.

Spotify then

Student Advice

"Watch videos before lecture"

"Watch the videos. Definitely helps with understanding the lecture beforehand, and the reason why I was so lost during the second half of the semester."

"Obviously watch the videos, try not to watch it at 2x speed (which is what I did and regret)...if you don't take time processing information, it will leave your brain by next morning"

"keep up with lectures, watch the videos, try to really understand everything along the way so it doesn't pile up at the end"

"Make sure to watch lecture videos before the lectures, so that lectures can be utilized for asking questions and further understanding."

Problem-Solving Practice

Solving problems becomes easier with **practice.**

Lab Monday/Tuesday: attendance is required (unless you're in mega lab)

Discussion on Wed/Thurs/Fri: attendance is required (unless you're in mega discussion)

These prepare you for weekly **homework** assignments & four larger programming **projects**

Quizzes during lab make sure that you can redo homework problems (with slight variations)

Drop-in one-on-one assignment help (called "**office hours**" at Cal) starts next week

What does a "discussion section" look like?

Expectation

https://engineering.berkeley.edu/students/academic-support/

Reality

https://www.microsoft.com/en-us/research/blog/grassroots-data-science-education-uc-berkeley/

Goal: Provide a great environment to learn how to solve problems through practice & *discussion*

Discussion (Starts This Week)

Unless you've elected the mega discussion...

- You should have a group number shared with around 5 students in your group, a room, and a discussion time. There are multiple groups per room.

What happens during discussion section?

- You're given an online worksheet full of problems to solve together.
- The point is to learn how to solve similar problems.
  - Discussion problems aren't graded; you don't have to solve them all.
  - If you solve them all but don't talk to anyone, you've missed out.
- Bring a laptop or tablet, but a phone or paper works in a pinch.

Student Advice

"Try to collaborate and with others and try to make friends within the class."

"Attend lab and make sure you understand how each program is structured"

"really utilize discussions. learning is easier with others!"

"Attend discussions and labs as it helps you discuss the content with other students"

"Go to discussion. I am so, so grateful for the fact that I had an active discussion group. Make the tough choice and commit yourself to spending that hour each week solving problems with other people just as confused as you are-- I guarantee that you'll look back on the decision and have no doubt that it was worthwhile."

"ASK OTHER PEOPLE FOR HELP WHEN NEEDED, DON'T BE AFRAID TO MAKE FRIENDS!!!"

Lab 0 Demo

Lab 0 sets up your computer to complete assignments.

- Python
- Visual Studio Code
- Two course-specific extensions designed so that you learn a lot from the assignments:
  - Preceptor – an AI helper that also checks your understanding
  - Provenance – tracks how you completed the assignment

(Demo)

Asking Questions

?

Ed: You can reach all staff (private posts) and all students (public posts)

kayo@berkeley.edu / denero@berkeley.edu: We might ask you to post on Ed

cs61a@berkeley.edu: Goes to several staff members & the instructors

Course Policies

Course Policies

Learning

Community

Course Staff

Details...

https://cs61a.org/fa26/syllabus/

Collaboration

Working together is highly encouraged!

What constitutes academic misconduct?

- *Please* don't look at someone else's code!
  Exceptions: lab, project partner, or **after you already solved the problem.**
- *Please* don't tell other people the answers! You can point them to what is wrong and describe how to fix it or show them a related example.
- *Please* don't use AI (such as ChatGPT or Copilot) to write answers for you.
  Exceptions: the AI-assisted part of the last project.

No Harassment or Discrimination

Disparaging remarks are unacceptable, especially targeting a gender, race, or ethnicity.

From the Berkeley Principles of Community:

> "We affirm the dignity of all individuals and strive to uphold a just community in which discrimination and hate are not tolerated."

From the EECS department mission:

> "Diversity, equity, and inclusion are core values in the Department of Electrical Engineering and Computer Sciences. Our excellence can only be fully realized by faculty, students, and staff who share our commitment to these values."

denero.org/feedback.html: If you want to stay anonymous but make me aware of something happening in the course.

EECS Student Climate & Incident Reporting Form: Informs the EECS department of any issues. You can also contact Susanne Kauer (skauer@berkeley.edu) directly.

Values, Operators, and Expressions

(Demo)

Types of expressions

An expression describes a computation and evaluates to a value

```
2          25 + 3          pi          'are you human'
```

Number or Numeral | Name | String

$$2^{100} \qquad \frac{6}{23} \qquad \sin \pi \qquad \log_2 1024$$

$$\sqrt{3493161} \qquad f(x) \qquad 7 \bmod 2$$

$$\lim_{x \to \infty} \frac{1}{x} \qquad \sum_{i=1}^{100} i \qquad |-1869| \qquad \binom{69}{18}$$

Call Expressions in Python

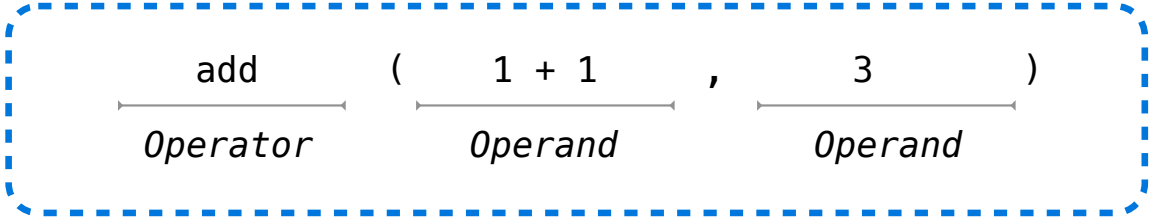
All expressions can be written as call expressions

(Demo)

Anatomy of a Call Expression

Expression: describes how to compute something, evaluates to a **value**

Call Expression



Operator and Operands are also expressions!

Evaluation Procedure for call expressions

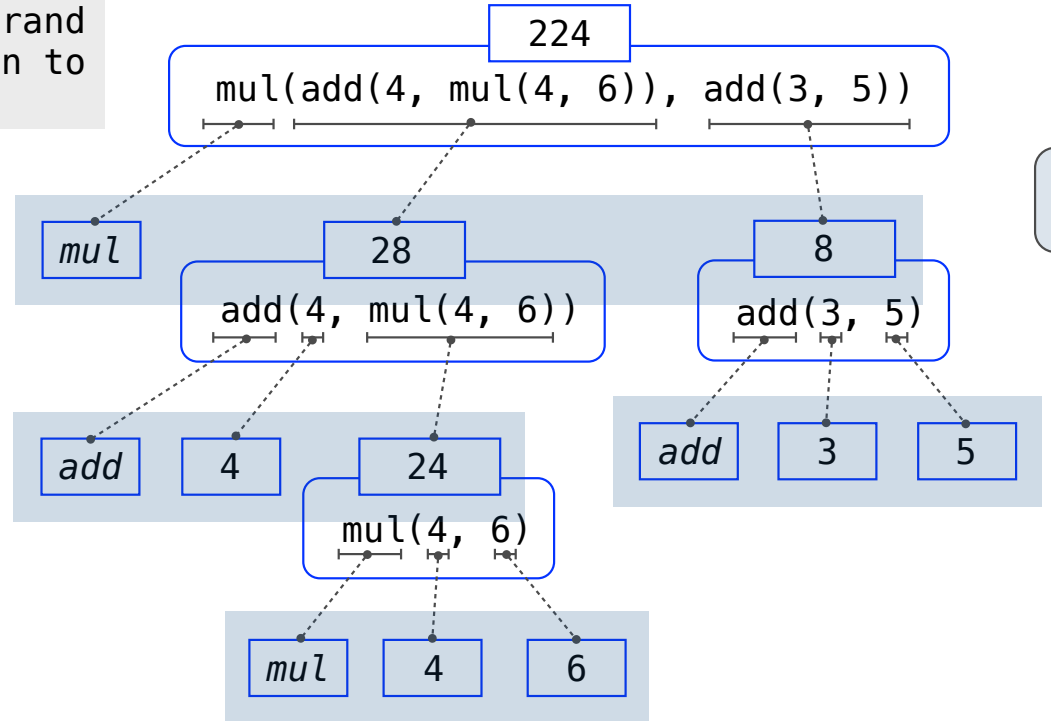
(1) Evaluate operator ⇩ **function**

(2) Evaluate each operand ⇩ **argument**

(3) **Apply** the **function** to the **arguments**

Evaluating Nested Expressions

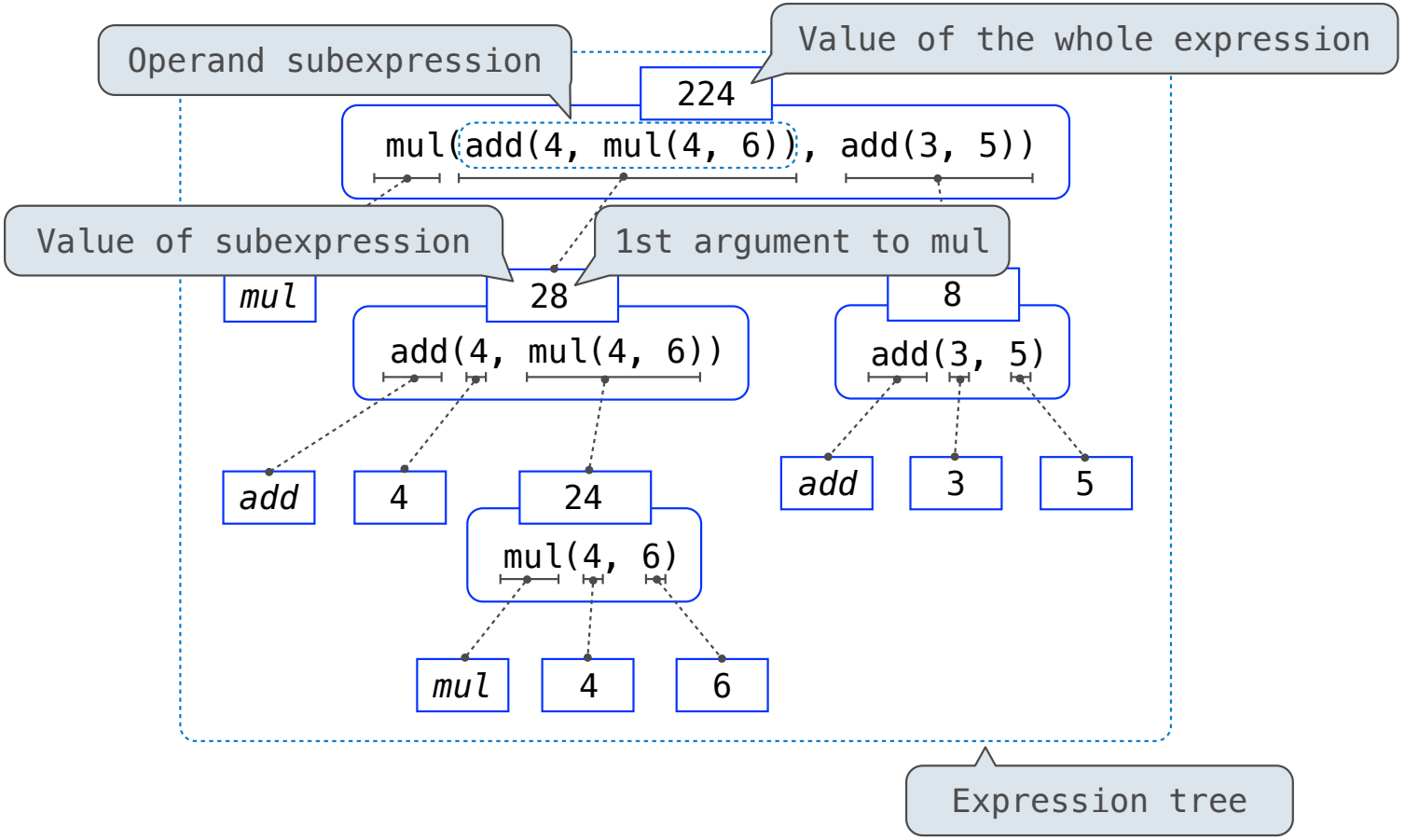
(1)Evaluate operator
(2)Evaluate each operand
(3)Apply the function to the arguments



Which part of this expression gets evaluated first?

pollev.com/cs61a

Evaluating Nested Expressions



Project Demo