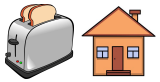
Welcome to Data C88C!

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:

- Monday 4pm–5pm,
  Undergraduate Academic Building 117

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:

- Wednesday 4pm–5pm,
  Undergraduate Academic Building 117

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

https://c88c.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

Different types of languages: Python & SQL

What about Artificial Intelligence?

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, so write programs on your own!

A course-specific AI tool is available to help if you get stuck.

AI-assisted programming is only taught in CS 61A.

c88c.org

How to Succeed

Lecture, Videos, and the Textbook

Videos posted to c88c.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 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

Problem-Solving Practice

Solving problems is an effective way to learn how to solve problems.

Lab: attendance is required (unless you're in the self-paced mega lab).

You'll discuss problems in groups, then work on the lab.

These prepare you for weekly **homework** assignments & 2 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.

This Week Only: If you didn't have lab (or didn't make it to lab), review the Discussion 0 handout and complete Lab 0 on your own.

What does a "discussion" look like (during your lab section)?

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*

Lab (Starts This Week)

Unless you've elected the mega lab...

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

What happens during the "discussion" at the start of lab?

- 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.

What happens after the discussion portion of lab?

- Work through the programming problems in the week's lab assignment. You're welcome to work with others and ask for help from your TA.

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

cs88@berkeley.edu: Goes to several staff members & the instructors

Should you take Data C88C?

According to the Syllabus: c88c.org/fa26/syllabus/

There is no formal programming-related prerequisite for Data C88C, but...

- Taking the course without any prior programming experience is typically quite challenging.
- Students who take the course without prior programming experience typically must spend more time to complete assignments.
- If you find it challenging to complete all of the required coursework in the first three weeks, we strongly recommend that you take another course first.

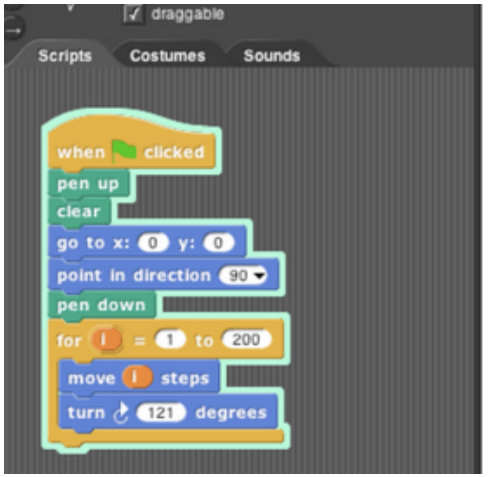
CS 10: The Beauty and Joy of Computing

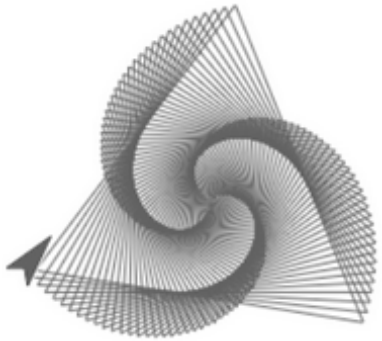
Designed for students without prior experience

A programming environment created by Berkeley, now used in courses around the world and online

An introduction to fundamentals (& Python) that sets students up for success in Data C88C and CS 61A

More info: http://cs10.org/





CS 61A

Data C88C is based on CS 61A, but covers only 3 out of 4 units worth of the content:

- Two programming projects (instead of four)
- Omits the unit on how web applications are designed and built, and a project where students write a web application using an LLM (both new this year)
- With this semester's updates to CS 61A, the extra unit in CS 61A is broadly relevant

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.

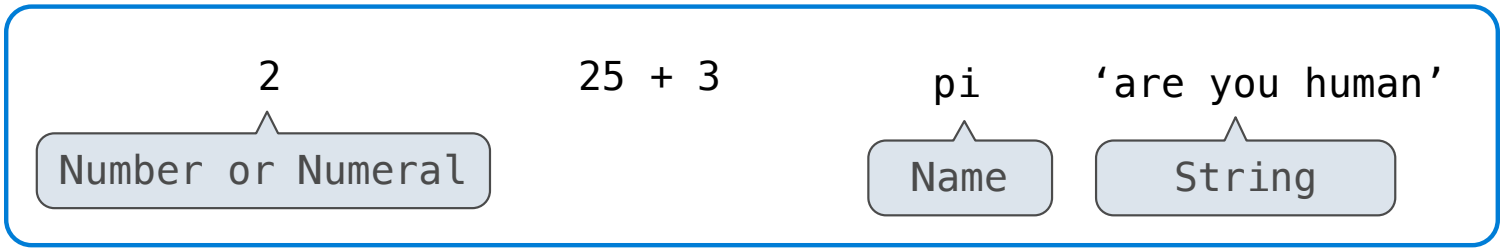
EECS Student Climate & Incident Reporting Form: Informs the EECS department of any issues. You can also contact Susanne Kauer (skauer@berkeley.edu) directly.

Values, Operators, and Expressions

(Demo)

Types of expressions

An expression describes a computation and evaluates to a value



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

$\sqrt{3493161}$

$f(x)$

$7 \bmod 2$ $\quad \lim_{x \to \infty} \frac{1}{x}$

$\sum_{i=1}^{100} i$ $\quad \binom{69}{18}$

$|-1869|$

Call Expressions in Python

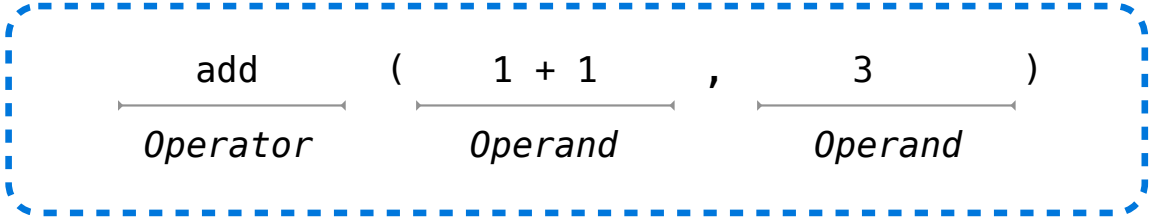
All expressions can be written as call expressions

(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

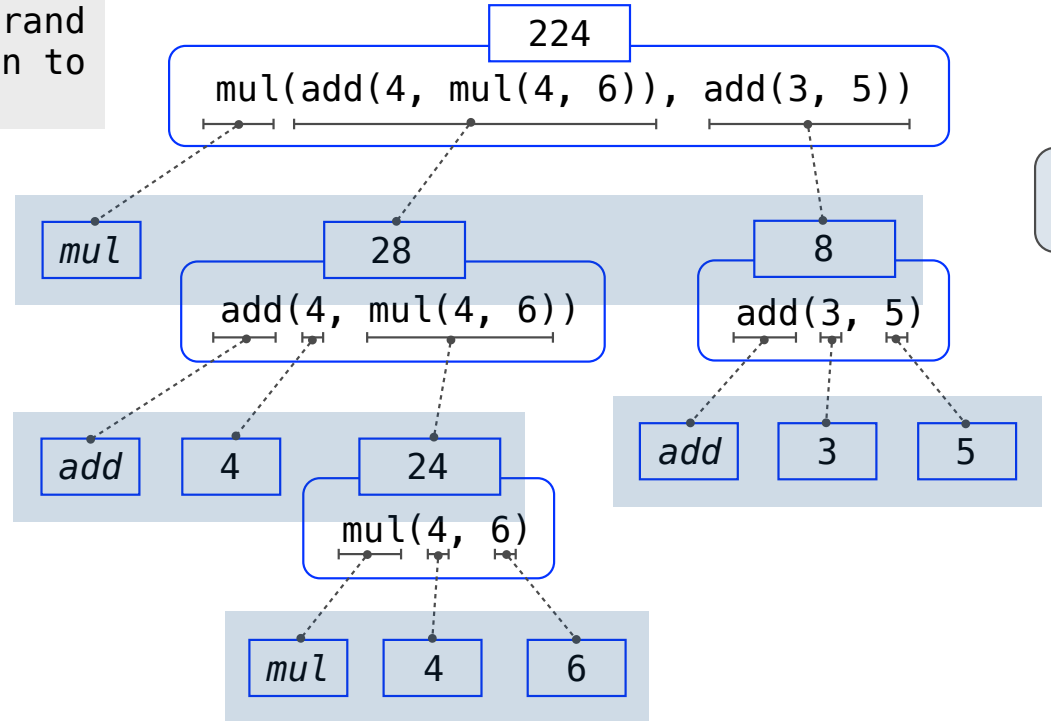
(1) Evaluate operator ⇩ **function**

(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/c88c

Evaluating Nested Expressions