

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)

#### Office hours start next week:

- Wednesdays 11–12 (781 Soda)
- Fridays 2:15–3:15 (781 Soda;  
follow me from lecture)

### John DeNero

denero@berkeley.edu

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

Research on machine translation  
and how people interact with AI

#### Office hours start next week:

- Mondays 2–3 (Location TBD)
- Wednesdays 11–12 (781 Soda)

<https://cs61a.org/staff/>

## About the Course

## What is This Course About?

---

A course about managing complexity

Mastering abstraction

Isolating and solving problems

Techniques for organizing complex programs

An introduction to programming

Full understanding of Python fundamentals

Large projects to demonstrate how to manage complexity

How computers interpret programming languages

Different types of languages: Python, Scheme, & SQL

[cs61a.org](http://cs61a.org)

How to Succeed

## Lecture, Videos, and the Textbook

---

Videos posted to [cs61a.org](http://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](http://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.



then



## Student Advice from Fall 2024

---

"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."

<https://cs61a.org/articles/advice/>

## 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**

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 the 4–6 students in your group, a room, and a discussion time. There will be 2–7 groups per room, so make sure you find the right group.
- TAs often oversee discussions in two rooms simultaneously.

What happens during discussion section?

- You're given an online worksheet full of problems to solve together & some instructions.
- The point is not just to solve those problems, but 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 works in a pinch.

## Student Advice from Fall 2024

---

"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!!!"

## Important: Lab 1

---

**Friday:** Because of the Labor Day holiday on Monday Sept 1, those of you assigned to Monday labs will have lab on **Friday 8/29 (this week only!).** You got an email about this.

**Prep:** Lab 0 is posted. Try to complete it **before** you come to Lab 1.

## 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/articles/about-61a/>

## Collaboration

---

### **Working together is highly encouraged**

- Discuss everything with each other; learn from your fellow students!
- Some projects can be completed with a partner

### **What constitutes academic misconduct?**

- Please don't look at someone else's code!  
Exceptions: lab, your 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 tools (such as ChatGPT or Copilot) to write code for you.
- Copying/generating project solutions causes people to fail the course.

### **Build good habits now**

## Let's Stop Harassment & Discrimination

---

Disparaging remarks targeting a particular gender, race, or ethnicity are not acceptable.

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."

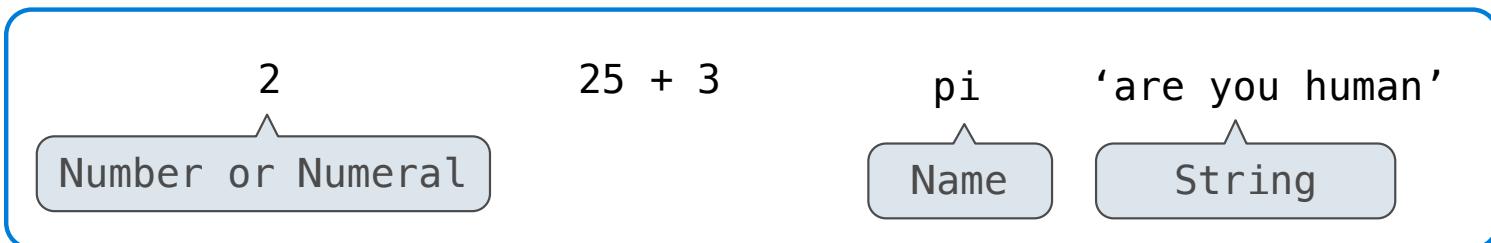
[denero.org/feedback.html](http://denero.org/feedback.html): If you want to stay anonymous but make me aware of something happening in the course.

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

## Values, Operators, and Expressions

## Types of expressions

An expression describes a computation and evaluates to a value



$$2^{100}$$

$$\frac{6}{23}$$

$$\sin \pi$$

$$\log_2 1024$$

$$7 \bmod 2$$

$$f(x)$$

$$|-1869|$$

$$\sum_{i=1}^{100} i$$

$$\sqrt{3493161}$$

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

$$\binom{69}{18}$$

## Call Expressions in Python

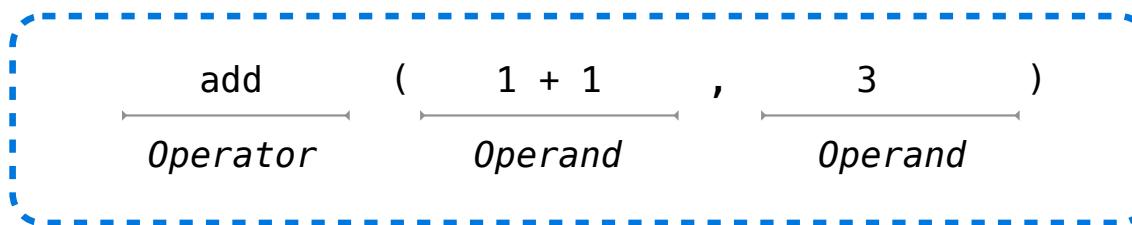
---

(Demo)

## Anatomy of a Call Expression

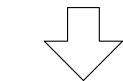
**Expression:** describes how to compute something,  
evaluates to a **value**

### Call Expression



Operator and Operands  
are also expressions!

(1) Evaluate operator



(2) Evaluate each operand

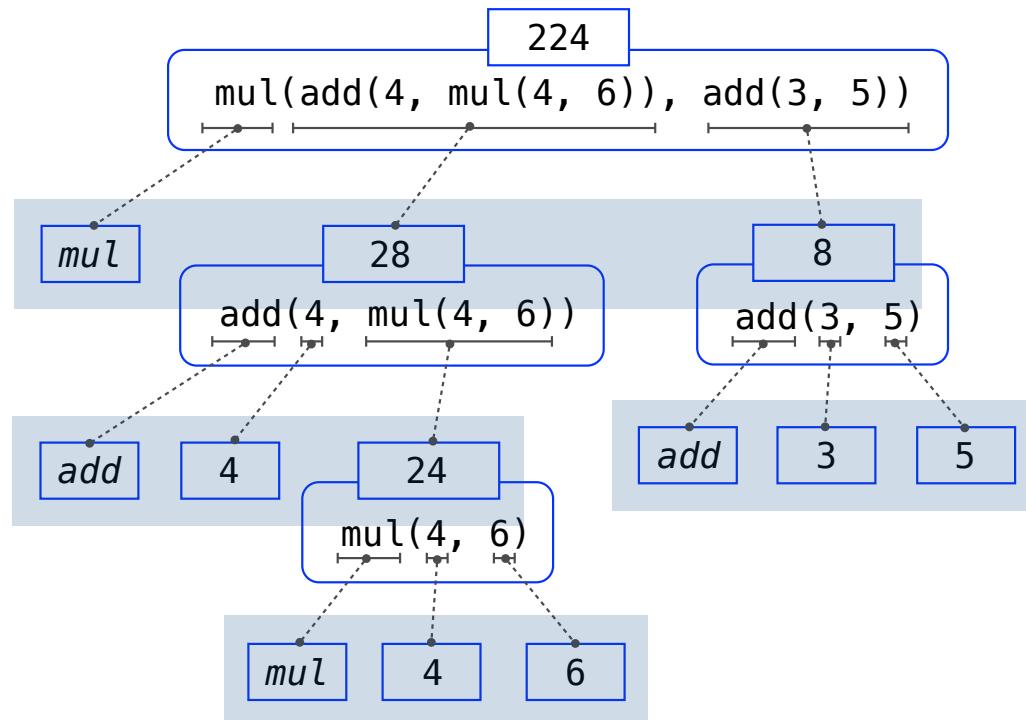


**Evaluation  
Procedure  
for call  
expressions**

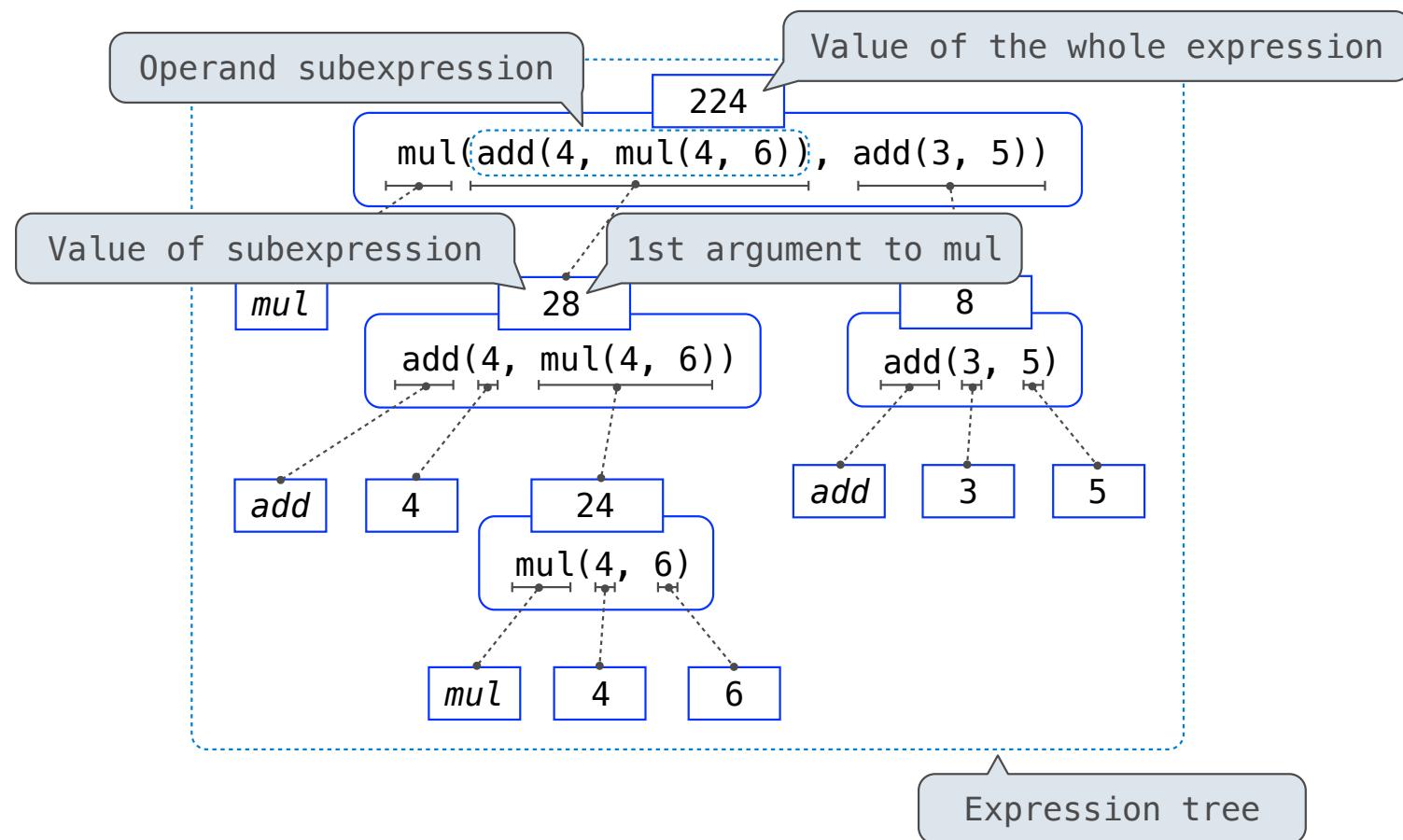
(3) **Apply** the **function** to the **arguments**

## Evaluating Nested Expressions

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



## Evaluating Nested Expressions



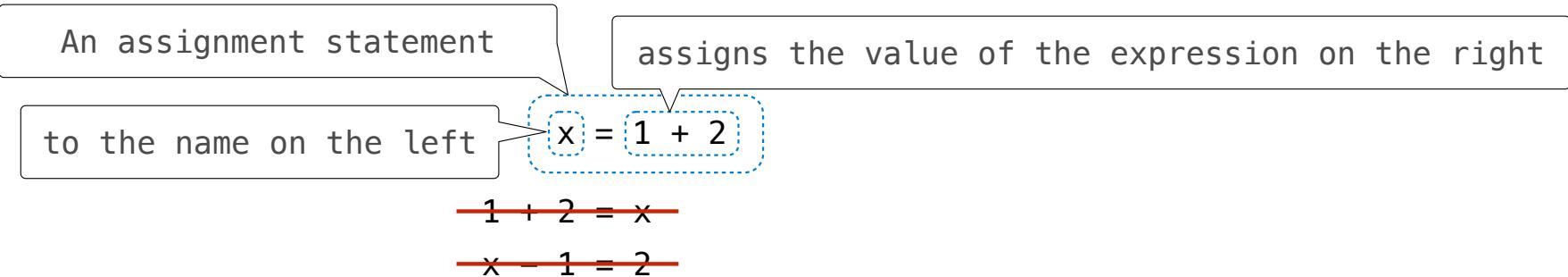
## Assignment Statements

## Assignment Statements

---

(Demo)

## Assignment Statements



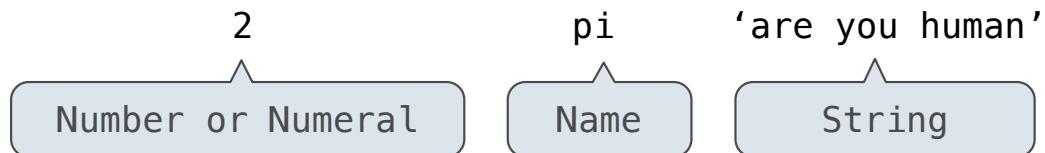
The expression (right) is evaluated, and its value is assigned to the name (left).

```
>>> x = 2
>>> y = x + 1
>>> y
3
>>> x = 5                               (Demo)
>>> x
5
>>> y
3
```

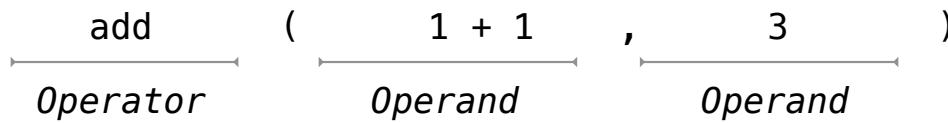
## Recap

**Expressions:** describe how to compute something, evaluate to a **value**

**Primitive expressions:**



**Call expressions:**



**Assignment:**

assigns the value of the expression on the right

to the name on the left

x = 1 + 2

## Projects You Will Build

## Projects You Will Build

---

(Demo)