

## Welcome to CS 61A

---

## Announcements

## About the Course

### The 61A Community

---

39 teaching assistants (TAs), known at Berkeley as GSIs or UGSIs:

- Teach lab & discussion sections
- Hold drop-in office hours
- Lots of other stuff: develop assignments, grade exams, etc.

Tutors:

- Hold drop-in office hours
- Teach 5-person mentoring sections
- Lots of other stuff: homework parties, mastery sections, etc.

Academic interns help answer individual questions during lab

1,450 fellow students make CS 61A unique

---

### Parts of the Course

---

**Lecture:** Videos posted to [cs61a.org](http://cs61a.org) before each live lecture

**Lab section:** The most important part of this course (*next week*)

**Discussion section:** The most important part of this course (*this week*)

**Staff office hours:** The most important part of this course (*next week*)

**Online textbook:** <http://composingprograms.com> (*read it before class*)

Weekly homework assignments, three exams, & four programming projects

Lots of optional special events to help you complete all this work


**Everything is posted to [cs61a.org](http://cs61a.org)**

---

## An Introduction to Computer Science

### What is Computer Science?

---

The study of  What problems can be solved using computation,  
How to solve those problems, and  
What techniques lead to effective solutions

Systems

Artificial Intelligence

Graphics

Security

Networking

Programming Languages

Theory

Scientific Computing

...

Decision Making

Robotics

Natural Language Processing

...

Answering Questions

Translation

...

### What is This Course About?

---

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

Different types of languages: Scheme & SQL

A challenging course that will demand a lot of you

---

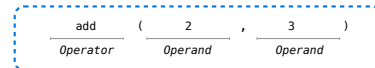




## Call Expressions in Python

All expressions can use function call notation  
(Demo)

## Anatomy of a Call Expression



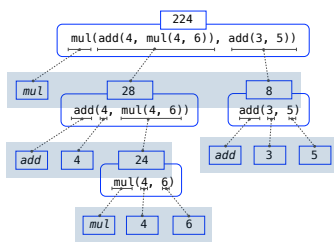
Operators and operands are also expressions

So they evaluate to values

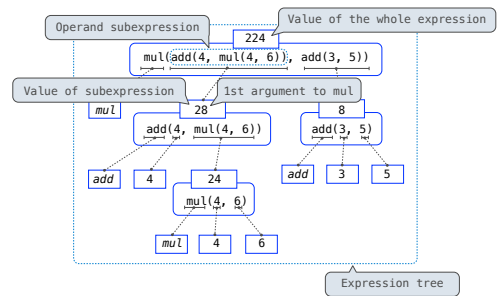
### Evaluation procedure for call expressions:

1. Evaluate the operator and then the operand subexpressions
2. Apply the function that is the value of the operator to the arguments that are the values of the operands

## Evaluating Nested Expressions



## Evaluating Nested Expressions



## Functions, Values, Objects, Interpreters, and Data

(Demo)