Functions

Welcome to CS 61A!

John DeNero

denero@berkeley.edu

Office hours in 781 Soda
(starting next week)

Wed 10am–11am & Thurs 10am–11am

By appointment: denero.org/meet.html

Fastest way to get answers: piazza.com/berkeley/spring2018/cs61a

Contact me & heads of staff: cs61a@berkeley.edu

The 61A Community

44 teaching assistants (TAs), formally known at Berkeley as UGSIs:

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

50+ mentors:

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

250+ academic interns help answer individual questions & check your progress

1,300+ fellow students make CS 61A unique

Parts of the Course

Lecture: Videos posted to 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

Weekly homework assignments, three exams, & four programming projects

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

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
  - Decision Making
  - Robotics
  - Natural Language Processing
    - Answering Questions
    - Translation
    - ...
  - ...
- Graphics
- Security
- Networking
- Programming Languages
- Theory
- Scientific Computing
- ...

What is This Course About?

A course about managing complexity

- Mastering abstraction
- Programming paradigms 编程范式

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

Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing

Designed for students without prior experience

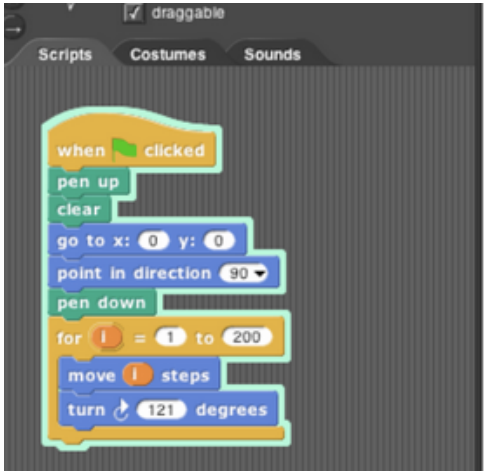
A programming environment created by Berkeley, now used in courses around the world and online

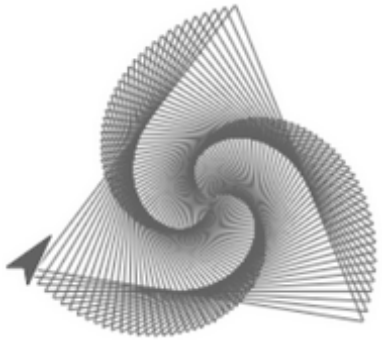
An introduction to fundamentals (& Python) that sets students up for success in CS 61A

Spring 2018: Dan Garcia

20+ person waitlist

More info: http://cs10.org/sp18/





Data Science 8: Foundations of Data Science

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

More statistics than computer science

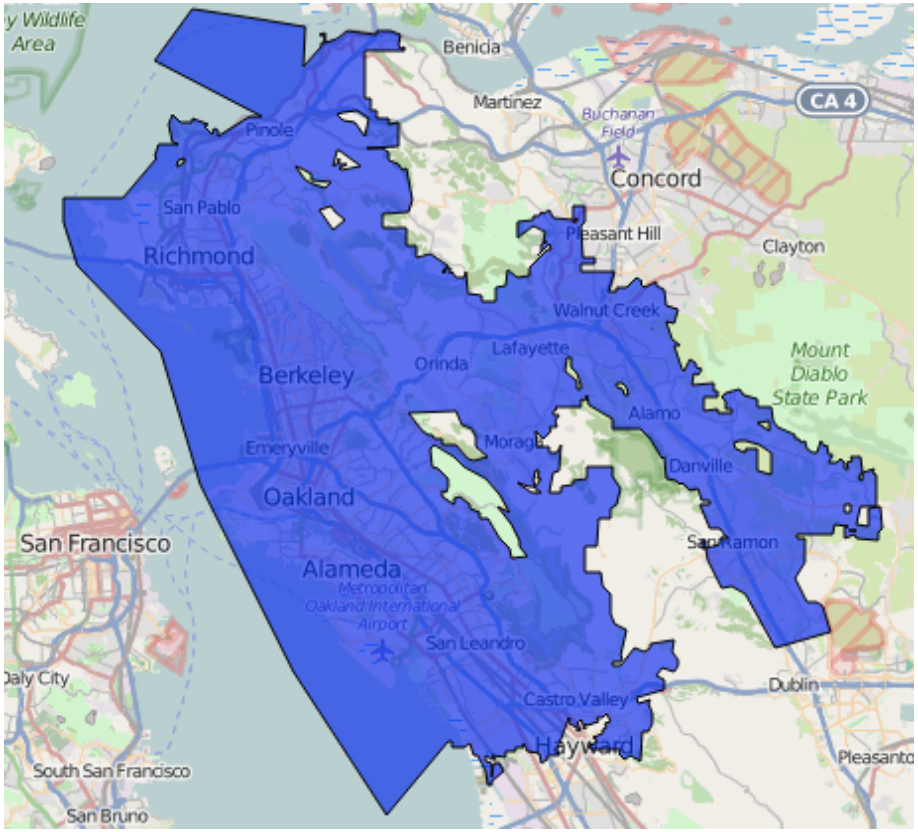
Great programming practice for CS 61A

Cross-listed as CS C8, Stat C8, & Info C8

Spring 2018: Ani Adhikari

100+ person waitlist

More info: http://data8.org/sp18



Course Policies

Learning

Community

Course Staff

Details...

http://cs61a.org/articles/about.html

Collaboration

Asking questions is highly encouraged

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

The limits of collaboration

- One simple rule: Don't share your code, except with your project partner
- Copying project solutions causes people to fail the course
- We really do catch people who violate the rules, because...
  - We also know how to search the web for solutions
  - We use computers to check your work

Build good habits now

Expressions

Types of expressions

What is an expression?

An expression describes a computation and evaluates to a value

$18 + 69$

$\frac{6}{23}$

$\sin \pi$

$\log_2 1024$

2^{100}

$f(x)$ f of x

$\sqrt{3493161}$

$7 \bmod 2$

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

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

$\binom{69}{18}$

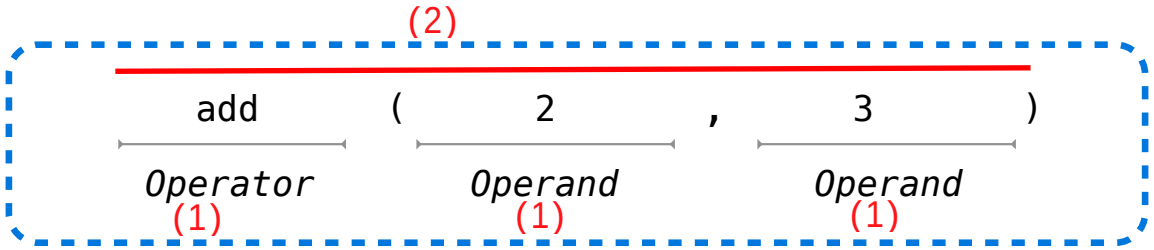
$|-1869|$

Call Expressions in Python

All expressions can use function call notation

(Demo)

Anatomy of a Call Expression



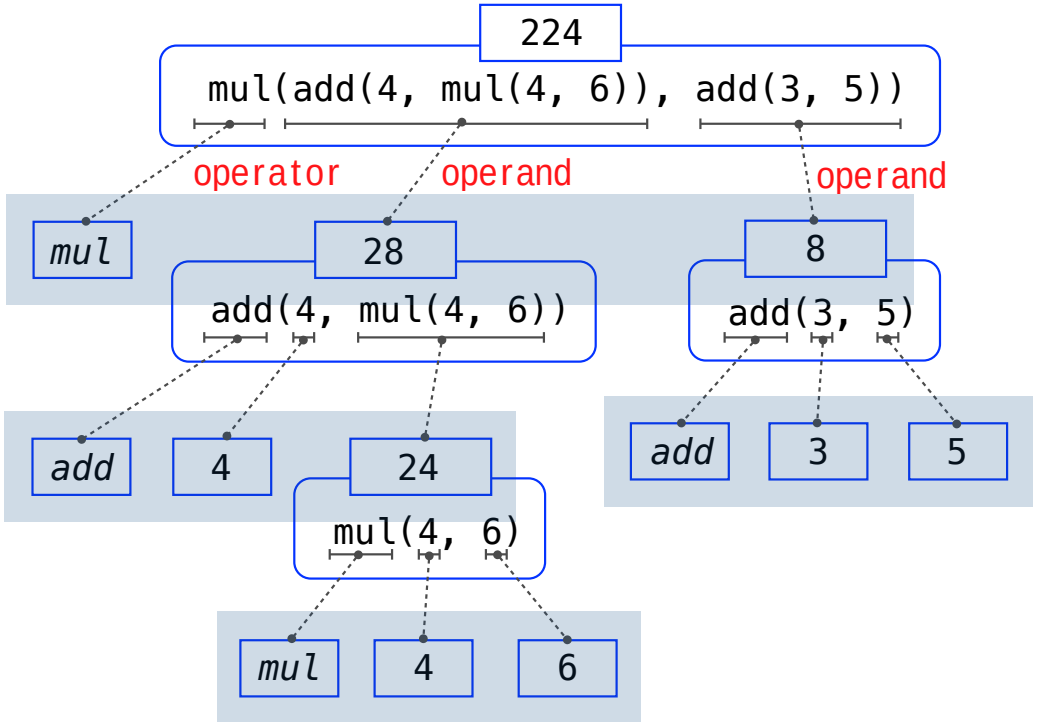
Operators and operands are also expressions

So they evaluate to values

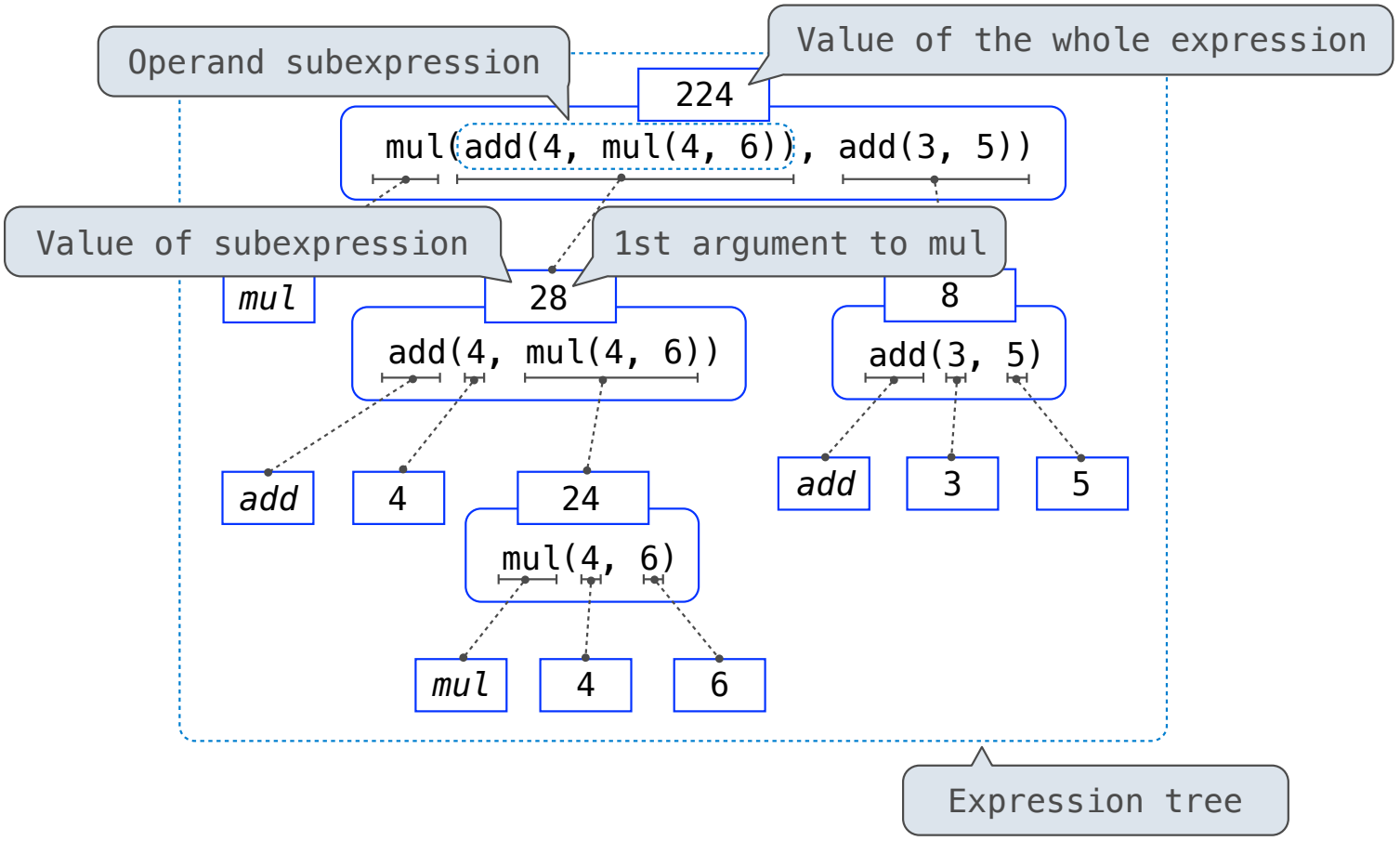
Evaluation procedure for call expressions:

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

Evaluating Nested Expressions



Evaluating Nested Expressions



```
# Objects
# Note: Download from http://composingprograms.com/shakespeare.txt
shakes = open('shakespeare.txt')
text = shakes.read().split()
len(text)
text[:25]
text.count('the')
text.count('thou')
text.count('you')
text.count('forsooth')
text.count(',')
```

Functions, Values, Objects, Interpreters, and Data

```
# Numeric expressions
2018
2000 + 18
1 - 2 + 3 + 4 * ((5 // 6) + 7 * 8 * 9)

# Call expressions
max(3, 4.5)
pow(100, 2)
pow(2, 100)
max(1, -2, 3, -4)
max(pow(10, 2), pow(2, 10), 1010)

# Importing and arithmetic with call expressions
from operator import add, mul
add(1, 2)
mul(4, 6)
mul(add(4, mul(4, 6)), add(3, 5))
add(2, mul(9, mul(add(4, mul(4, 6)), add(3, 5))))
```

(Demo)

```
# Sets
words = set(text)
len(words)
max(words)
max(words, key=len)

# Reversals
'draw'[::-1]
{w for w in words if w == w[::-1] and len(w)>4}
{w for w in words if w[::-1] in words and len(w) == 4}
{w for w in words if w[::-1] in words and len(w) > 6}
```