

Welcome to CS 61A

Announcements

About the Course

The 61A Community

The 61A Community

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

The 61A Community

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

- Teach lab & discussion sections

The 61A Community

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

- Teach lab & discussion sections
- Hold drop-in office hours

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.

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:

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

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

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.

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

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

Parts of the Course

Lecture: Videos posted to `cs61a.org` before each live lecture

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*)

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*)

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*)

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> (*read it before class*)

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> (*read it before class*)

Weekly homework assignments, three exams, & four programming projects

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> (*read it before class*)

Weekly homework assignments, three exams, & four programming projects

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

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> (*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`

An Introduction to Computer Science

What is Computer Science?

What is Computer Science?

The study of


What is Computer Science?

The study of

What problems can be solved using computation,

What is Computer Science?

The study of



What problems can be solved using computation,
How to solve those problems, and

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

What is Computer Science?

The study of Systems

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

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

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

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

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

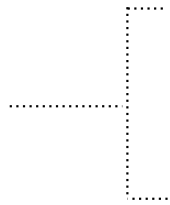
Programming Languages

Theory

Scientific Computing

...

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

...

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

Graphics

Security

Networking

Programming Languages

Theory

Scientific Computing

...

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

Graphics

- Robotics

Security

Networking

Programming Languages

Theory

Scientific Computing

...

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

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

...

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

...

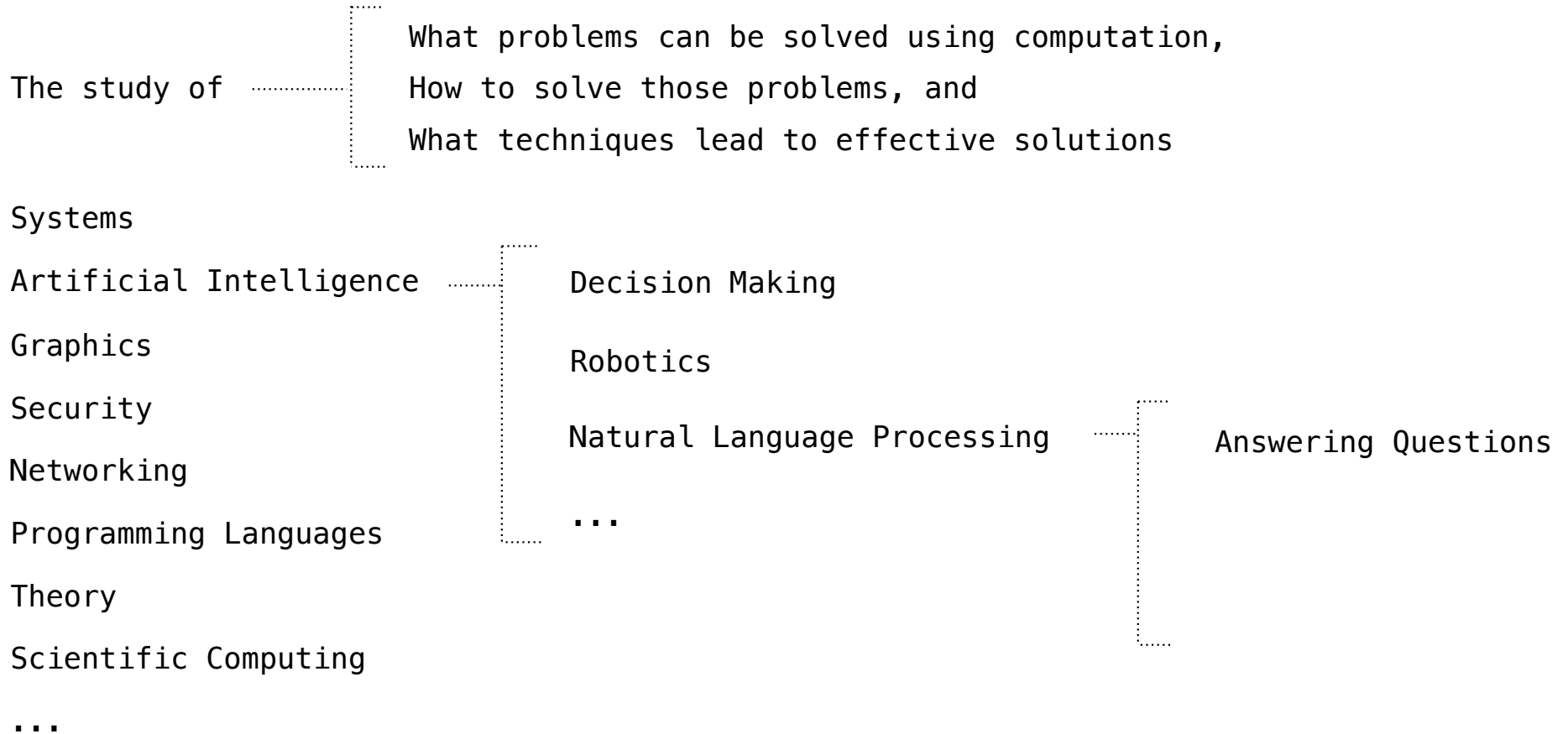
Decision Making

Robotics

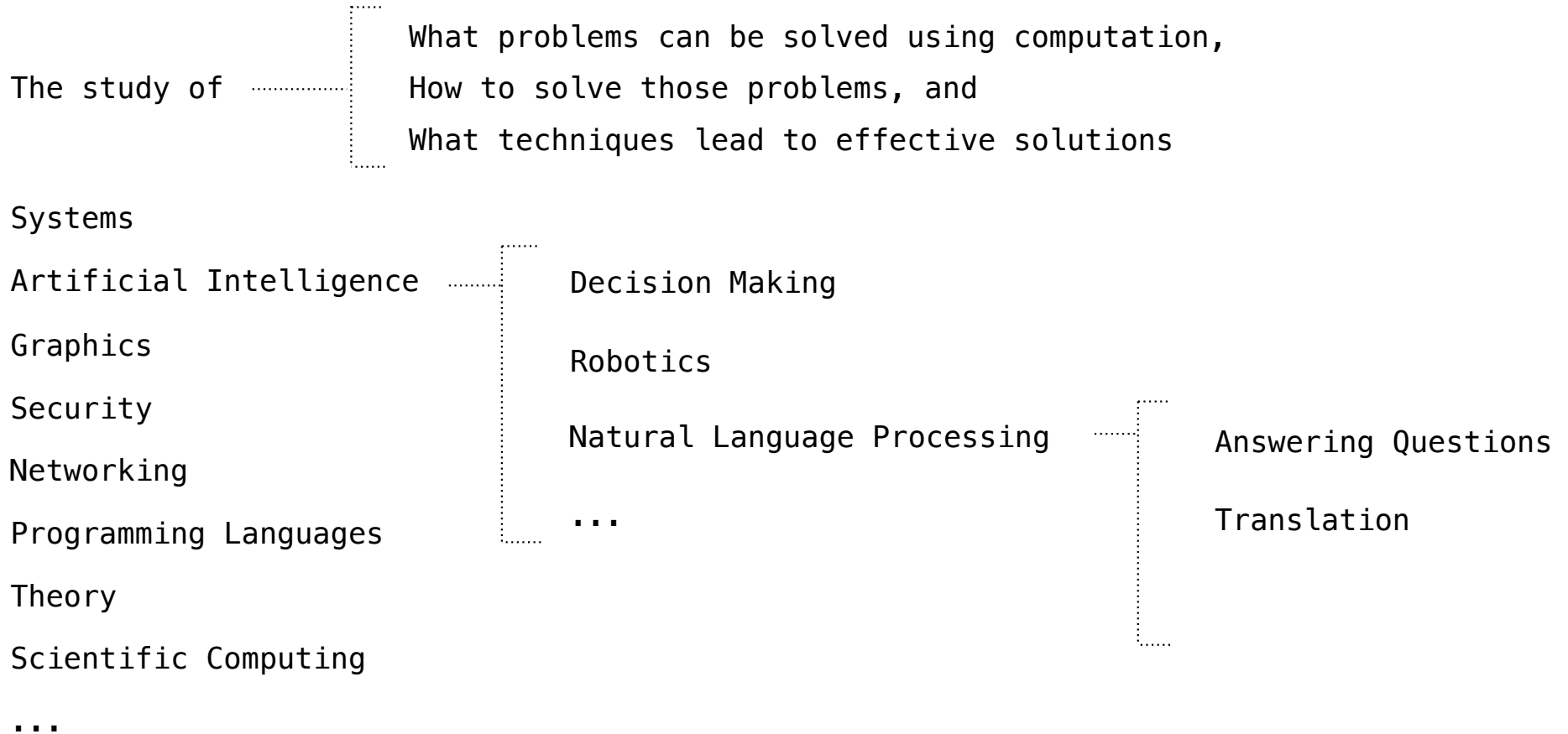
Natural Language Processing

...

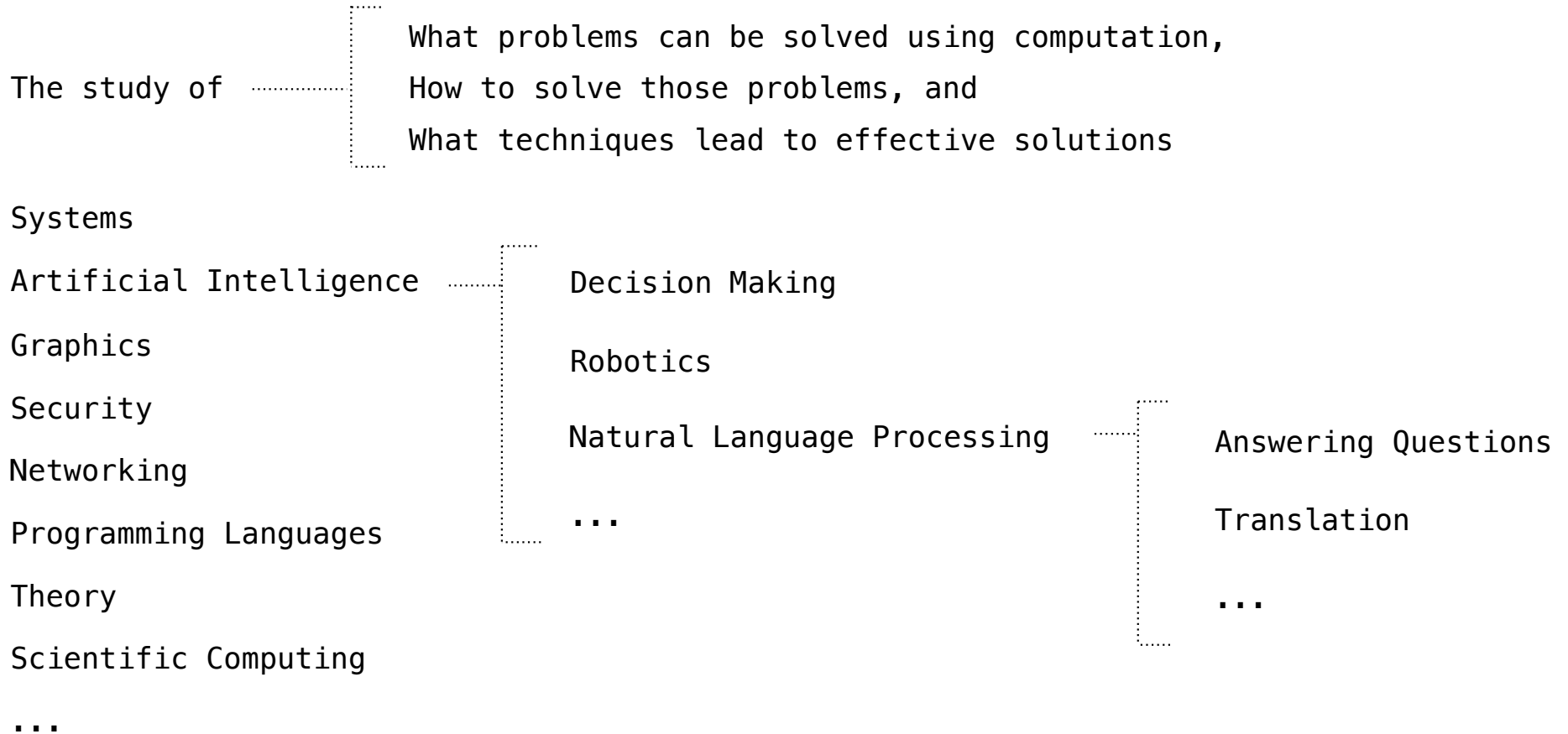
What is Computer Science?



What is Computer Science?



What is Computer Science?



What is This Course About?

What is This Course About?

A course about managing complexity

What is This Course About?

A course about managing complexity

Mastering abstraction

What is This Course About?

A course about managing complexity

Mastering abstraction

Programming paradigms

What is This Course About?

A course about managing complexity

Mastering abstraction

Programming paradigms

An introduction to programming

What is This Course About?

A course about managing complexity

Mastering abstraction

Programming paradigms

An introduction to programming

Full understanding of Python fundamentals



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



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



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



λ



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



Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing

CS 10: The Beauty and Joy of Computing



CS 10: The Beauty and Joy of Computing



CS 10: The Beauty and Joy of Computing

Designed for students without prior experience



CS 10: The Beauty and Joy of Computing

Designed for students without prior experience

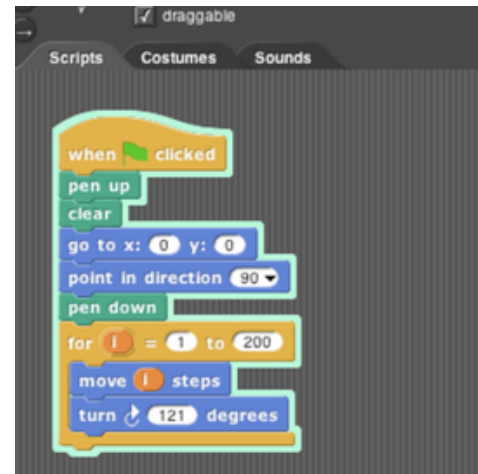
A programming environment created by Berkeley,
now used in courses around the world and online



CS 10: The Beauty and Joy of Computing

Designed for students without prior experience

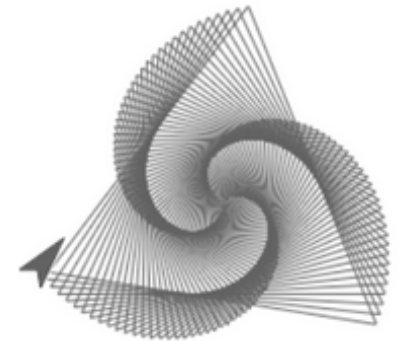
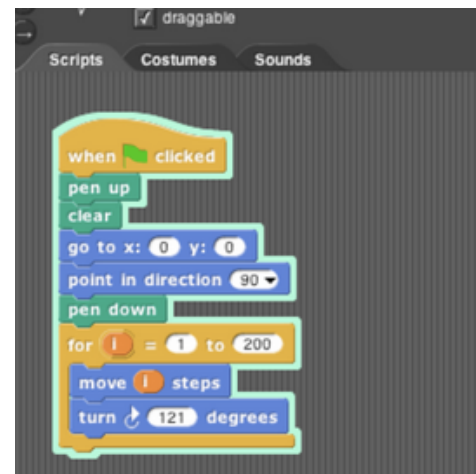
A programming environment created by Berkeley,
now used in courses around the world and online



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

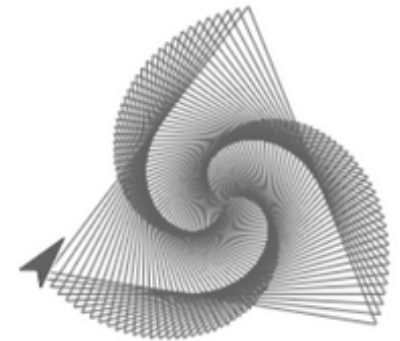
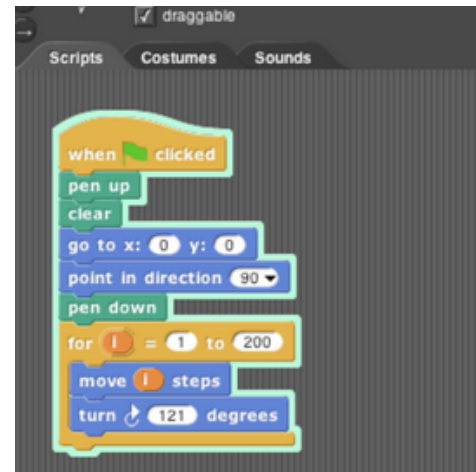


CS 10: The Beauty and Joy of Computing

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



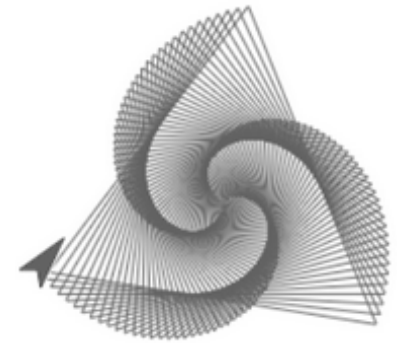
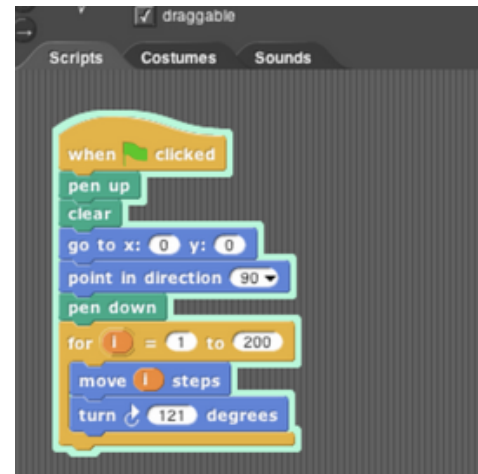
CS 10: The Beauty and Joy of Computing

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 2020: Dan Garcia



CS 10: The Beauty and Joy of Computing

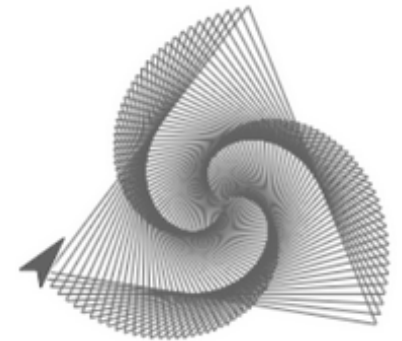
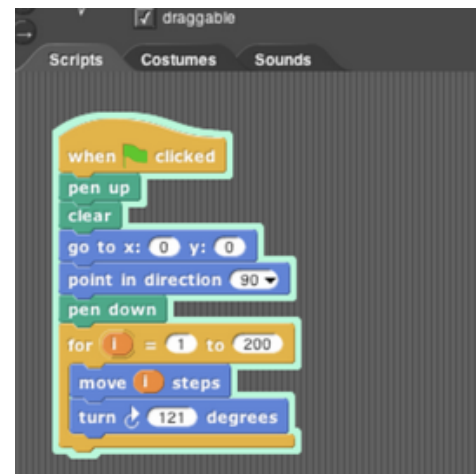
Designed for students without prior experience

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 2020: Dan Garcia

11 open seats (as of Wed 1/22)



CS 10: The Beauty and Joy of Computing

Designed for students without prior experience

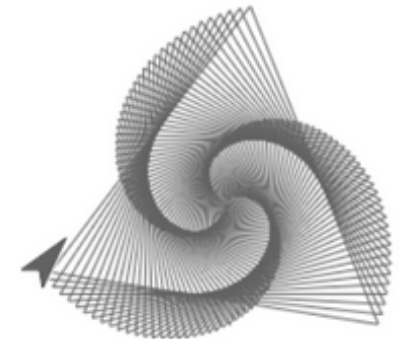
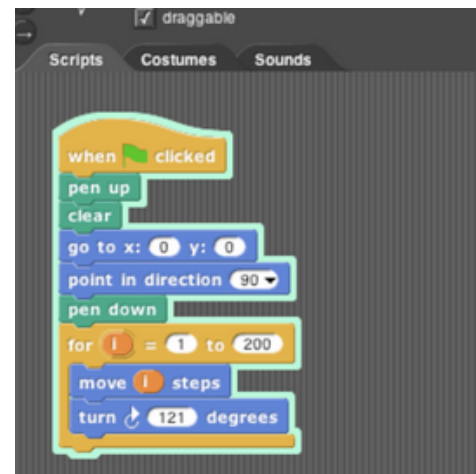
A programming environment created by Berkeley,
now used in courses around the world and online

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

Spring 2020: Dan Garcia

11 open seats (as of Wed 1/22)

Monday & Wednesday 3–4 in 120 Latimer



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

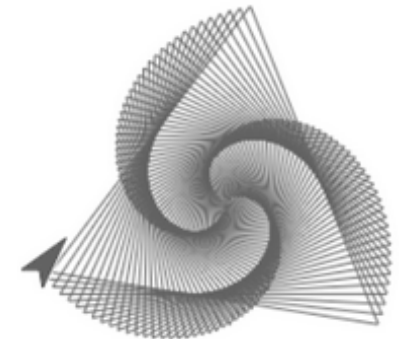
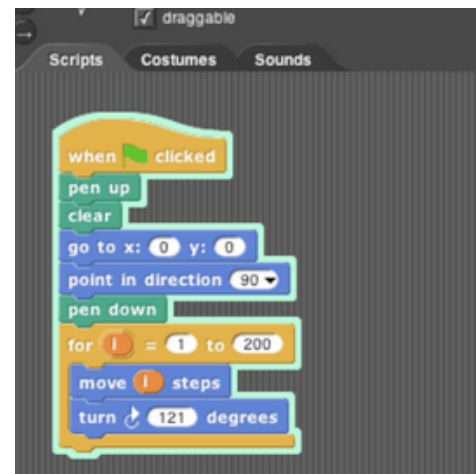
An introduction to fundamentals (& Python)
that sets students up for success in CS 61A

Spring 2020: Dan Garcia

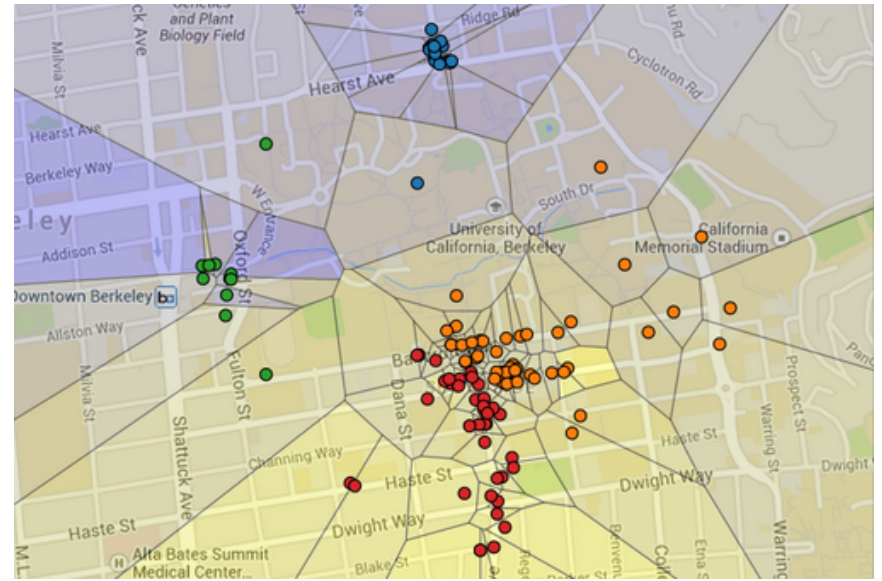
11 open seats (as of Wed 1/22)

Monday & Wednesday 3–4 in 120 Latimer

More info: <http://cs10.org/>

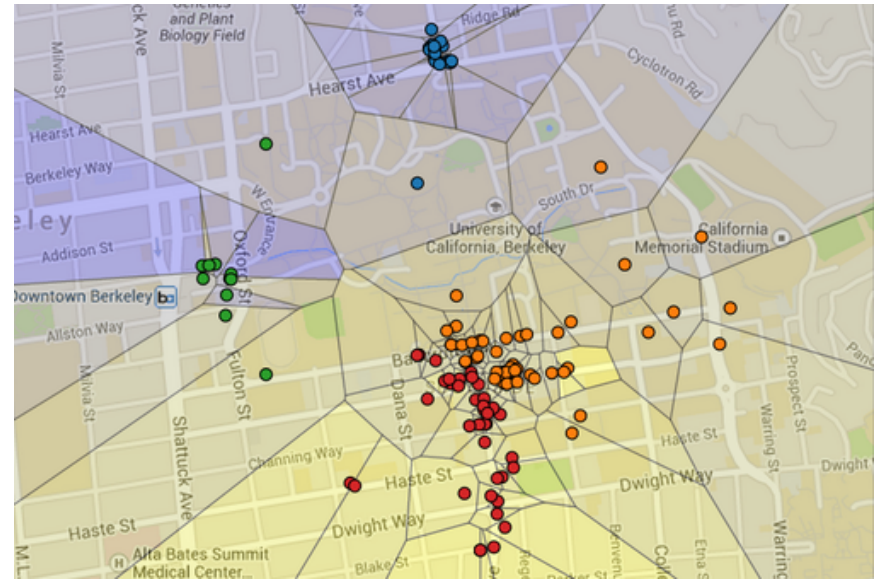


CS 88: Computational Structures in Data Science



CS 88: Computational Structures in Data Science

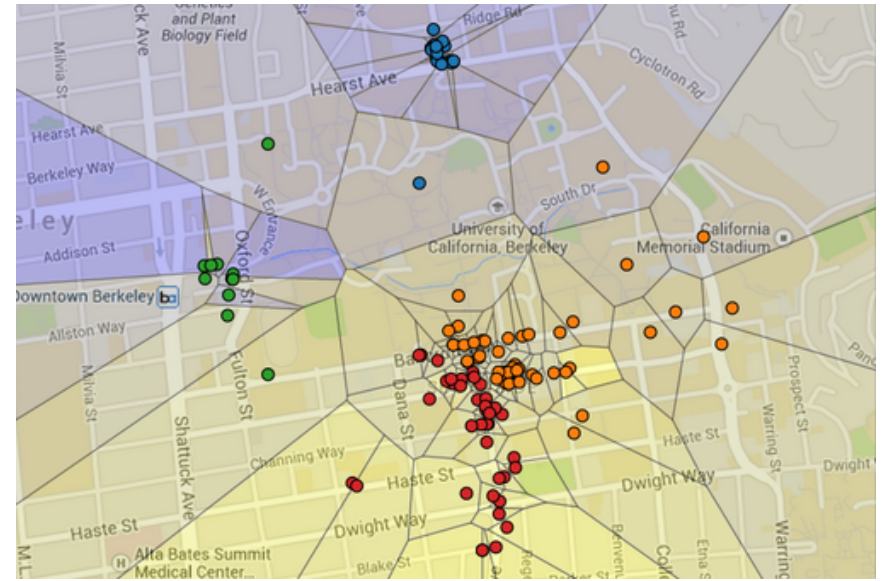
Alternative to CS 61A with very similar content



CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

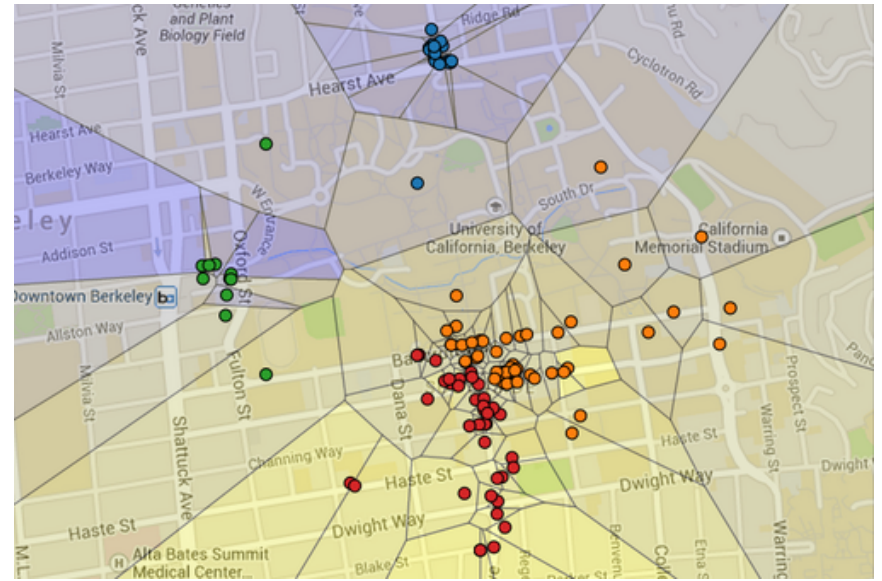
- Data 8 overlaps with ~25% of CS 61A



CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

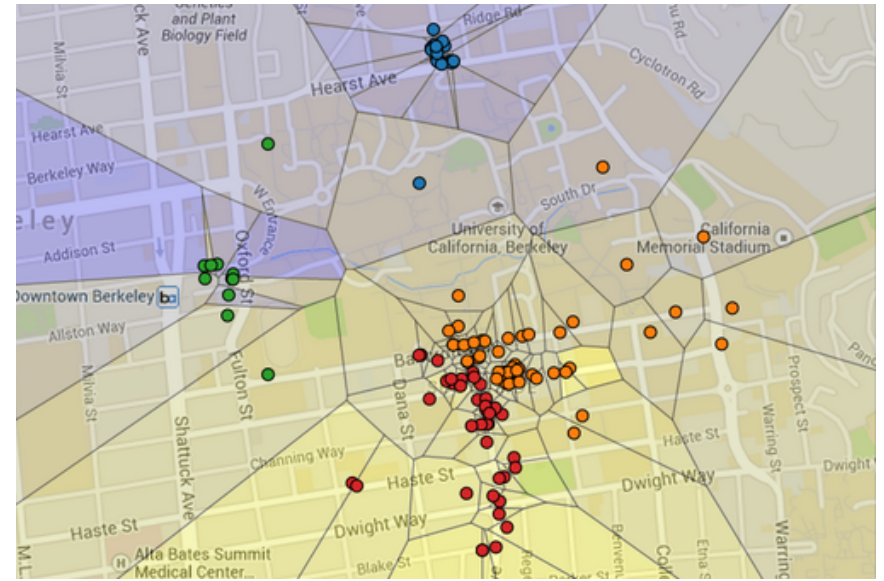


CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

Both together cover ~75% of CS 61A, enough to skip CS 61A and go directly to CS 61B



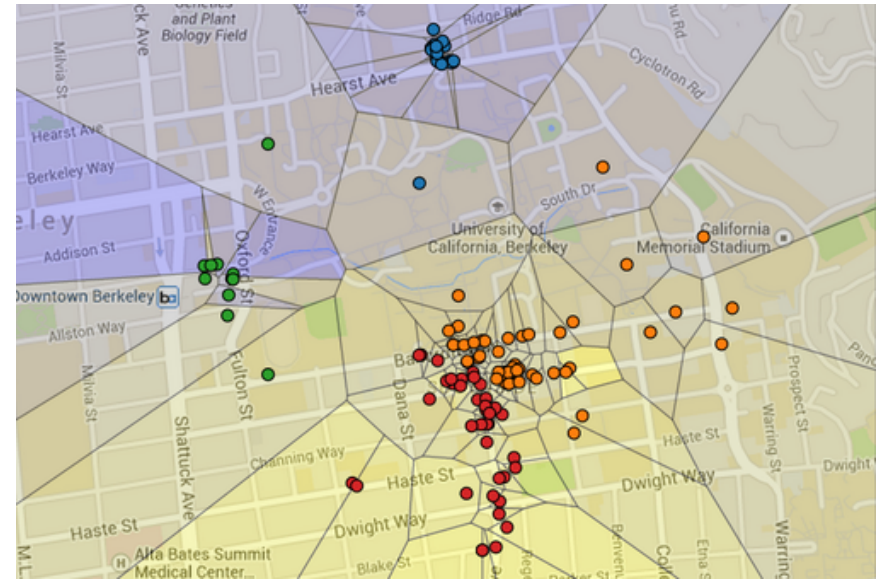
CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

Both together cover ~75% of CS 61A, enough to skip CS 61A and go directly to CS 61B

Some students take CS 61A after CS 88 for a very thorough introduction to programming



CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

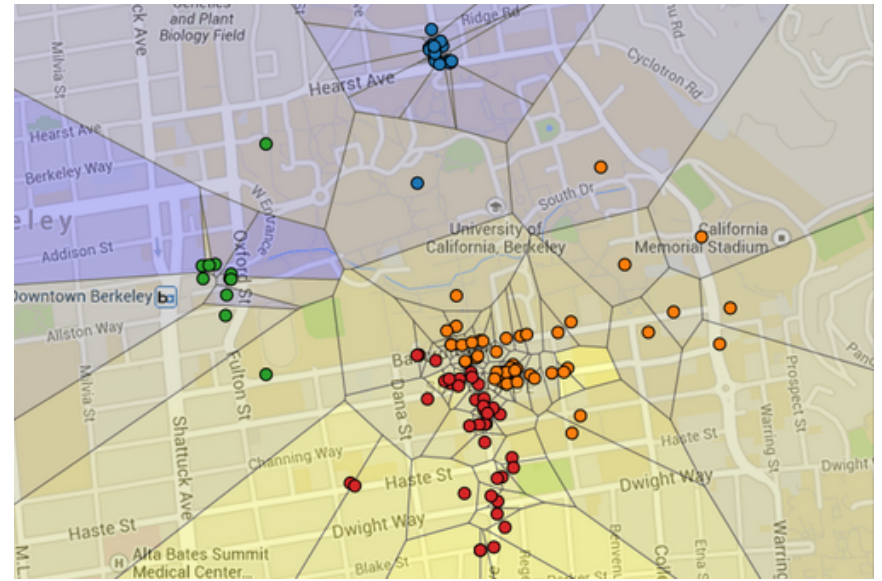
- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

Both together cover ~75% of CS 61A, enough to skip CS 61A and go directly to CS 61B

Some students take CS 61A after CS 88 for a very thorough introduction to programming

Spring 2020: 3 units and a revised syllabus

80 open seats (as of Wed 1/22)



CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

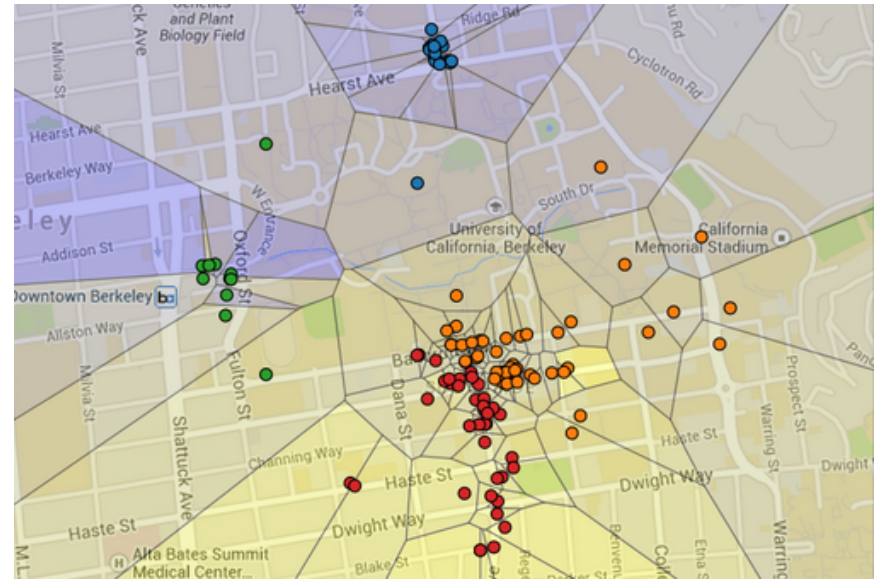
Both together cover ~75% of CS 61A, enough to skip CS 61A and go directly to CS 61B

Some students take CS 61A after CS 88 for a very thorough introduction to programming

Spring 2020: 3 units and a revised syllabus

80 open seats (as of Wed 1/22)

Monday & Friday 1–2 in 10 Evans



CS 88: Computational Structures in Data Science

Alternative to CS 61A with very similar content

- Data 8 overlaps with ~25% of CS 61A
- CS 88 overlaps with 50% of CS 61A or more

Both together cover ~75% of CS 61A, enough to skip CS 61A and go directly to CS 61B

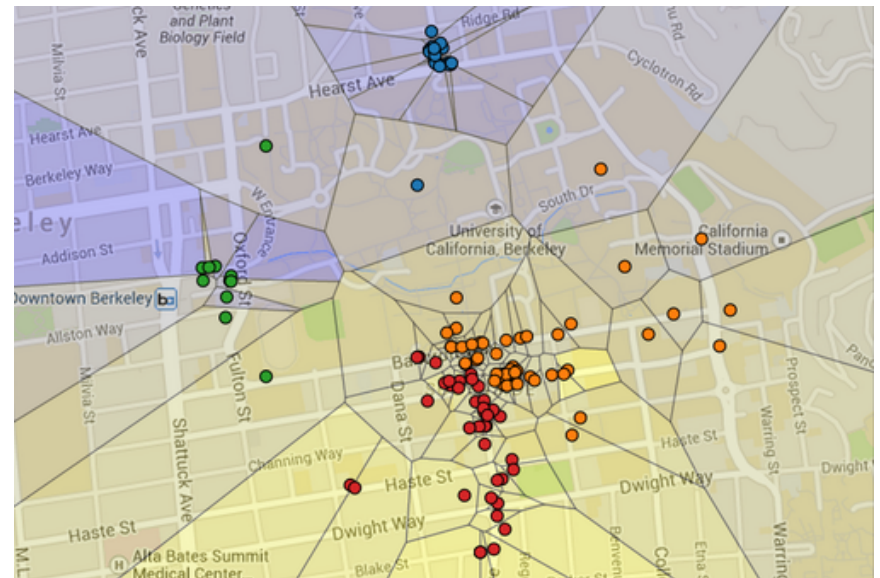
Some students take CS 61A after CS 88 for a very thorough introduction to programming

Spring 2020: 3 units and a revised syllabus

80 open seats (as of Wed 1/22)

Monday & Friday 1-2 in 10 Evans

More info: <https://cs88-website.github.io/>



Course Policies

Course Policies

Learning

Learning
Community

Learning

Community

Course Staff

Learning
Community
Course Staff

Details...

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

Collaboration

Collaboration

Asking questions is highly encouraged

Collaboration

Asking questions is highly encouraged

- Discuss everything with each other; learn from your fellow students!

Collaboration

Asking questions is highly encouraged

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

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

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

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

- *Please* don't look at someone else's code!
Exceptions: lab, your project partner, or **after** you already solved the problem

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

- *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, but don't tell them what to type, and don't type for them

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

- *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, but don't tell them what to type, and don't type for them
- Copying project solutions causes people to fail the course

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

- *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, but don't tell them what to type, and don't type for them
- Copying project solutions causes people to fail the course
- We really do catch people who violate the rules, and we're getting even better at it.

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

- *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, but don't tell them what to type, and don't type for them
- Copying project solutions causes people to fail the course
- We really do catch people who violate the rules, and we're getting even better at it.

Build good habits now

Expressions

Types of expressions

Types of expressions

An expression describes a computation and evaluates to a value

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sqrt{3493161}$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\sqrt{3493161}$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\sqrt{3493161}$$

$$| - 1869 |$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\sqrt{3493161}$$

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

$$|-1869|$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\sqrt{3493161}$$

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

$$|-1869|$$

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

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$f(x)$$

$$\sqrt{3493161}$$

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

$$|-1869|$$

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

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$2^{100}$$

$$f(x)$$

$$\sqrt{3493161}$$

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

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

$$|-1869|$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\log_2 1024$$

$$2^{100}$$

$$f(x)$$

$$\sqrt{3493161}$$

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

$$|-1869|$$

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

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\log_2 1024$$

$$2^{100}$$

$$f(x)$$

$$\sqrt{3493161}$$

$$7 \bmod 2$$

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

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

$$|-1869|$$

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\log_2 1024$$

$$2^{100}$$

$$f(x)$$

$$\sqrt{3493161}$$

$$7 \bmod 2$$

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

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

$$|-1869|$$

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

Types of expressions

An expression describes a computation and evaluates to a value

$$18 + 69$$

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

$$\sin \pi$$

$$\log_2 1024$$

$$2^{100}$$

$$f(x)$$

$$\sqrt{3493161}$$

$$7 \bmod 2$$

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

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

$$|-1869|$$

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

Call Expressions in Python

All expressions can use function call notation
(Demo)

Anatomy of a Call Expression

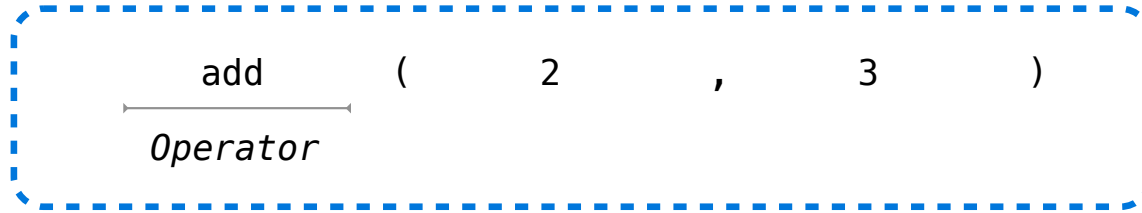
Anatomy of a Call Expression

add (2 , 3)

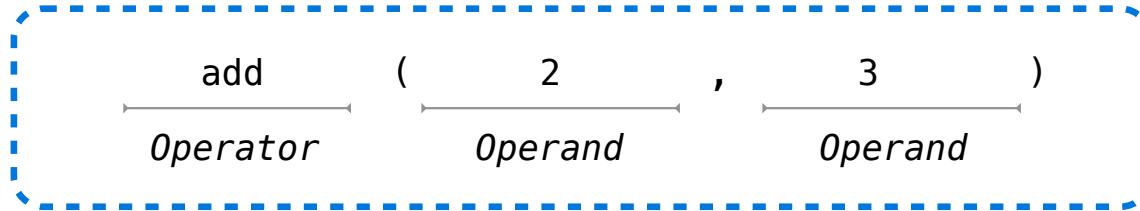
Anatomy of a Call Expression

add (2 , 3)

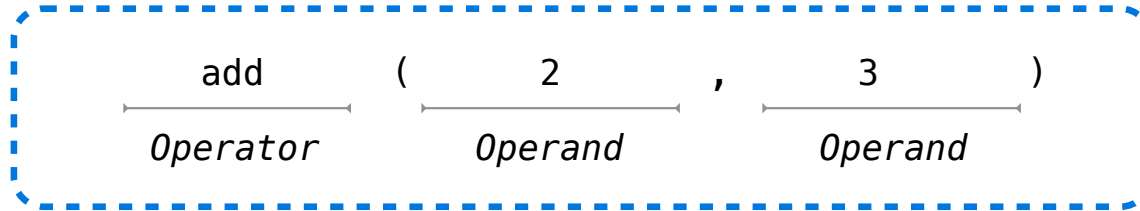
Anatomy of a Call Expression



Anatomy of a Call Expression

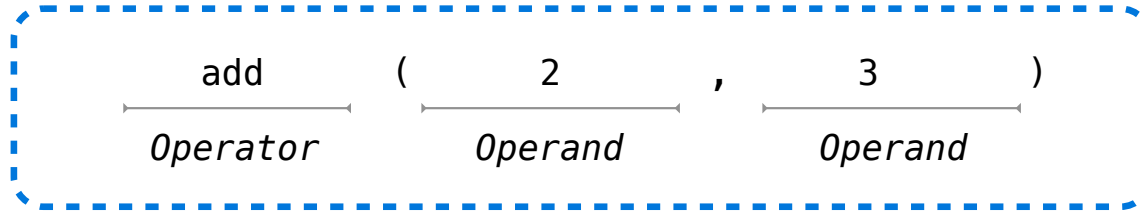


Anatomy of a Call Expression



Operators and operands are also expressions

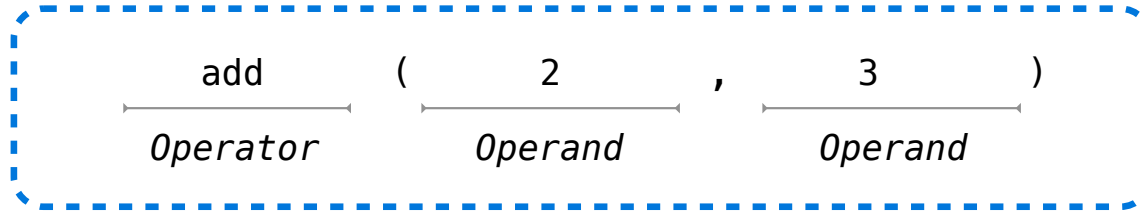
Anatomy of a Call Expression



Operators and operands are also expressions

So they evaluate to values

Anatomy of a Call Expression

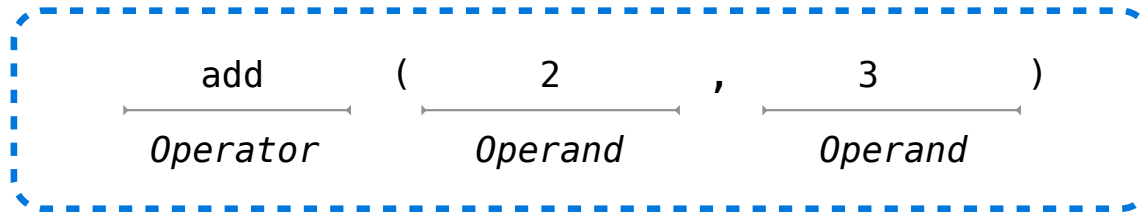


Operators and operands are also expressions

So they evaluate to values

Evaluation procedure for call expressions:

Anatomy of a Call Expression



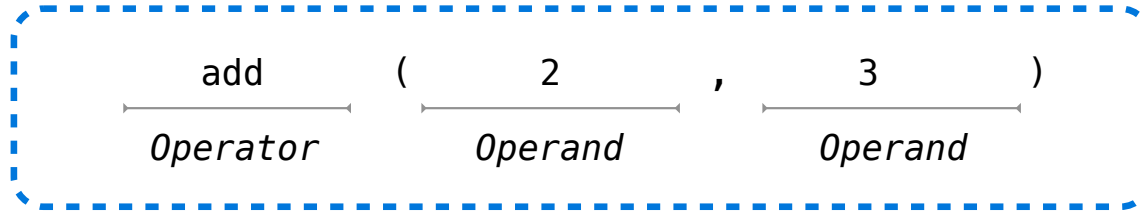
Operators and operands are also expressions

So they evaluate to values

Evaluation procedure for call expressions:

1. Evaluate the operator and then the operand subexpressions

Anatomy of a Call Expression



Operators and operands are also expressions

So they evaluate to values

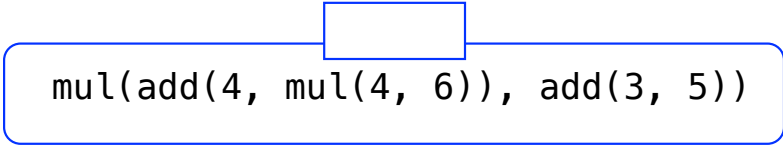
Evaluation procedure for call expressions:

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

```
mul(add(4, mul(4, 6)), add(3, 5))
```

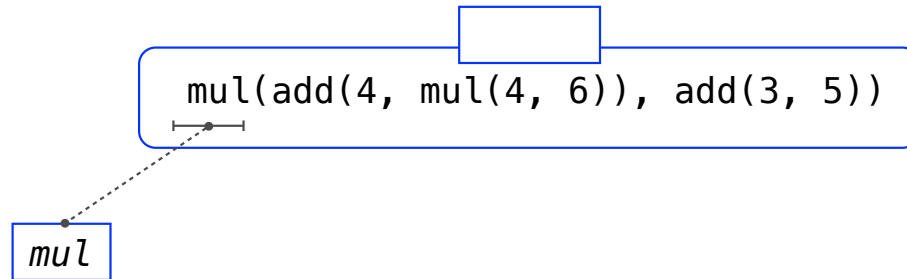

Evaluating Nested Expressions



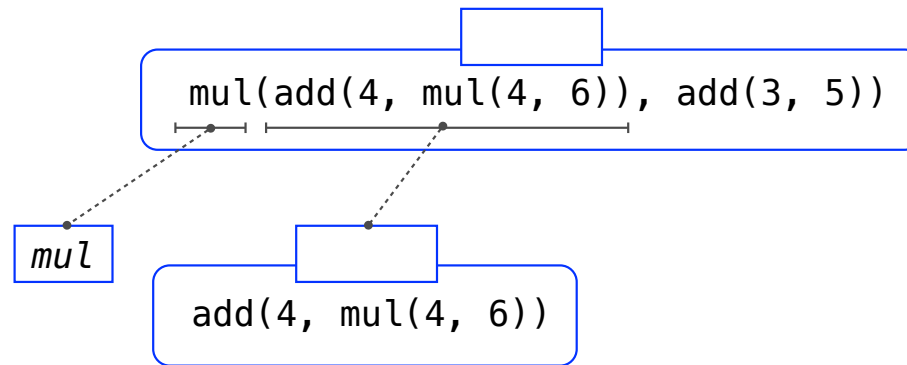
The diagram shows the expression `mul(add(4, mul(4, 6)), add(3, 5))` enclosed in a rounded rectangular box. A smaller rectangular box is positioned above the `mul(4, 6)` sub-expression, with a vertical line connecting it to the `mul(4, 6)` part of the main expression, indicating the current step in the evaluation process.

```
mul(add(4, mul(4, 6)), add(3, 5))
```

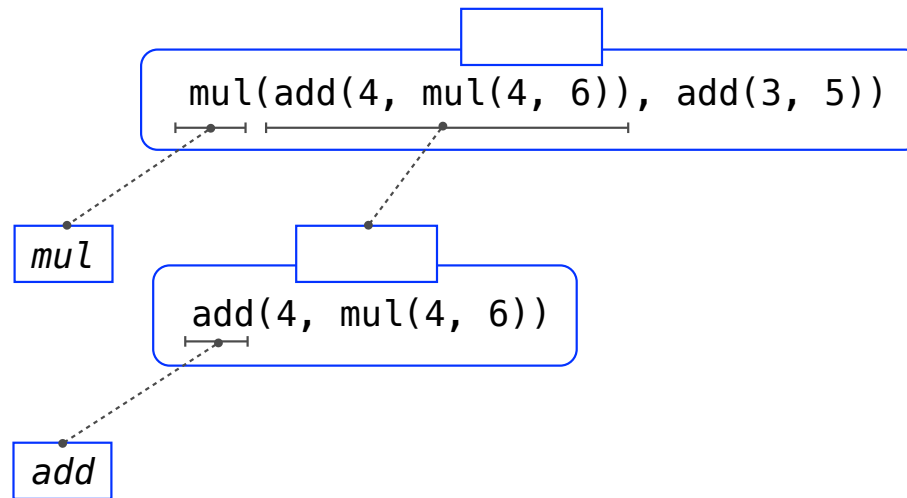
Evaluating Nested Expressions



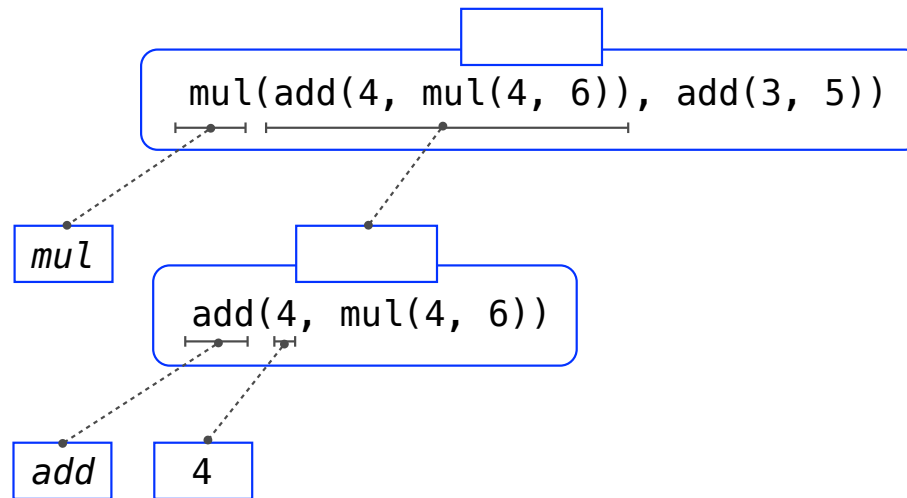
Evaluating Nested Expressions



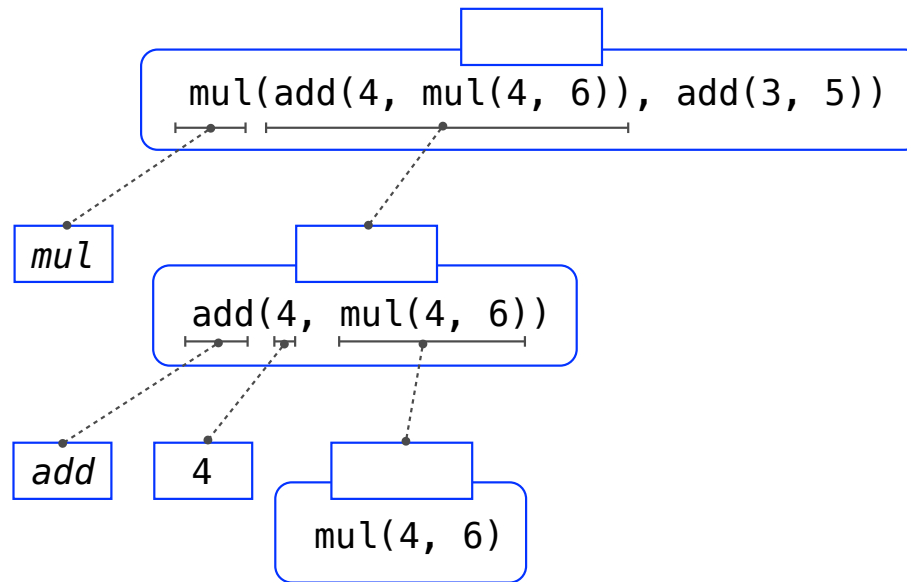
Evaluating Nested Expressions



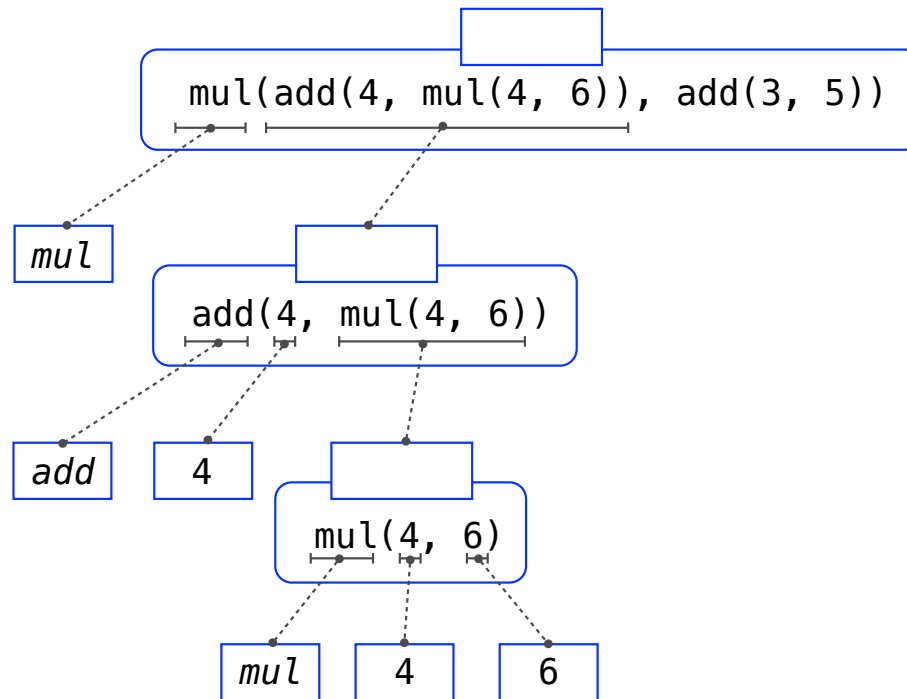
Evaluating Nested Expressions



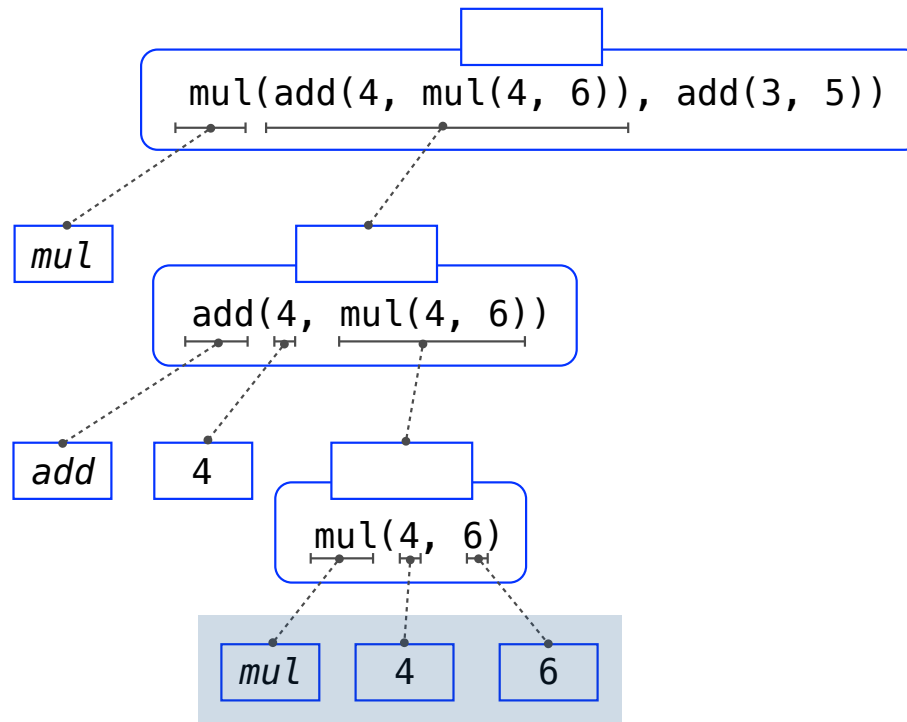
Evaluating Nested Expressions



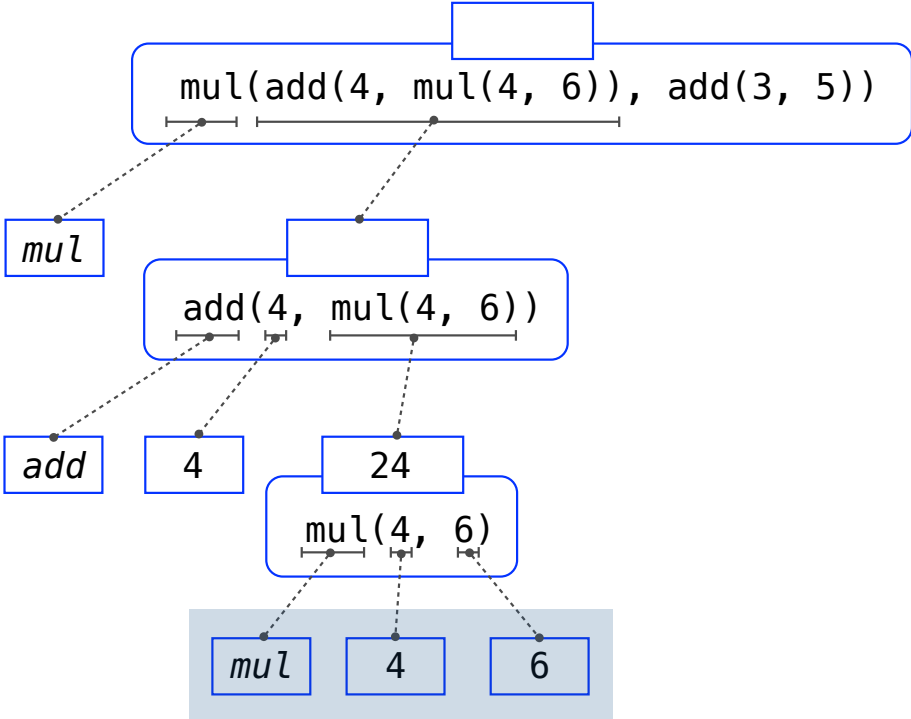
Evaluating Nested Expressions



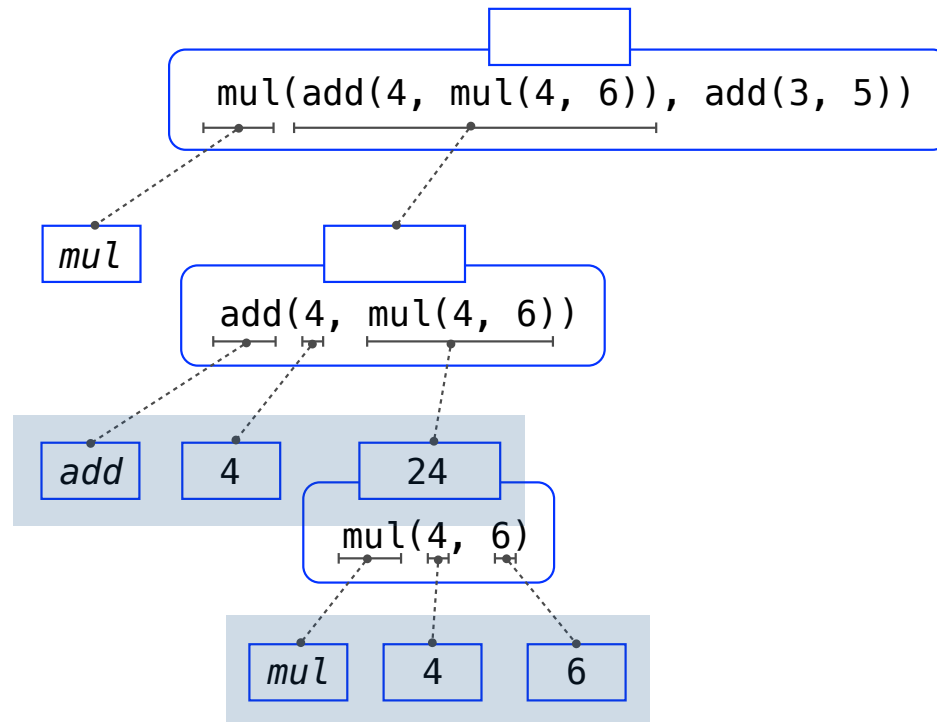
Evaluating Nested Expressions



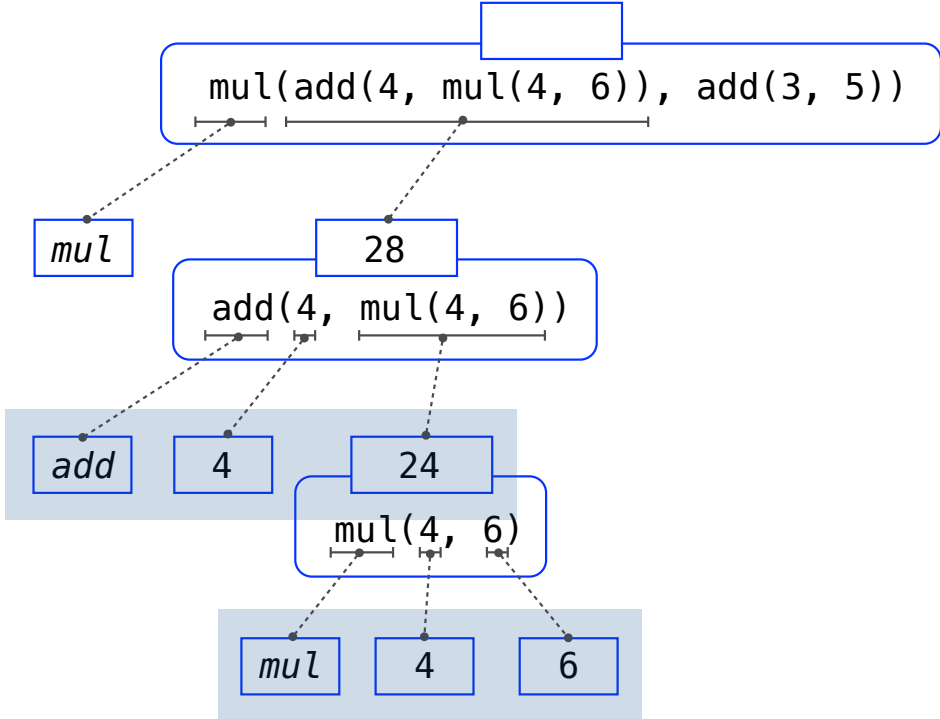
Evaluating Nested Expressions



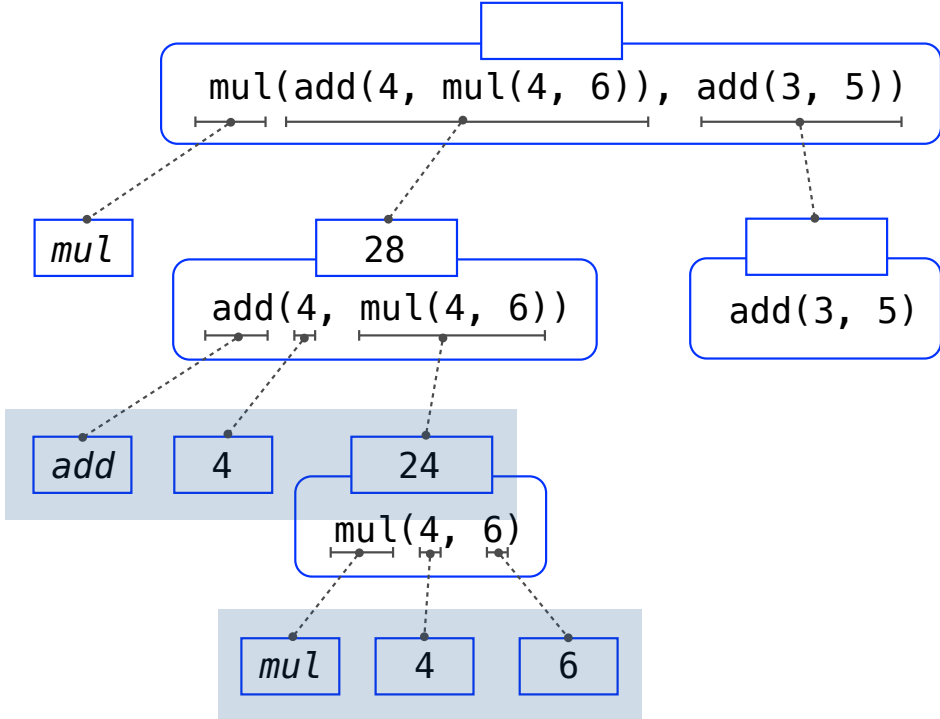
Evaluating Nested Expressions



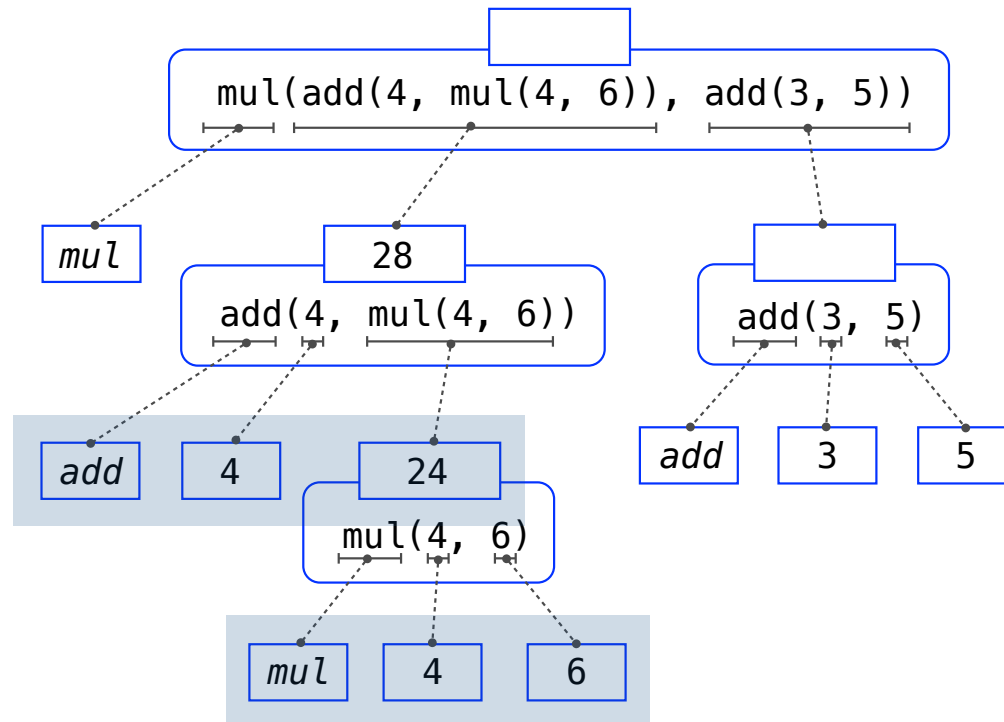
Evaluating Nested Expressions



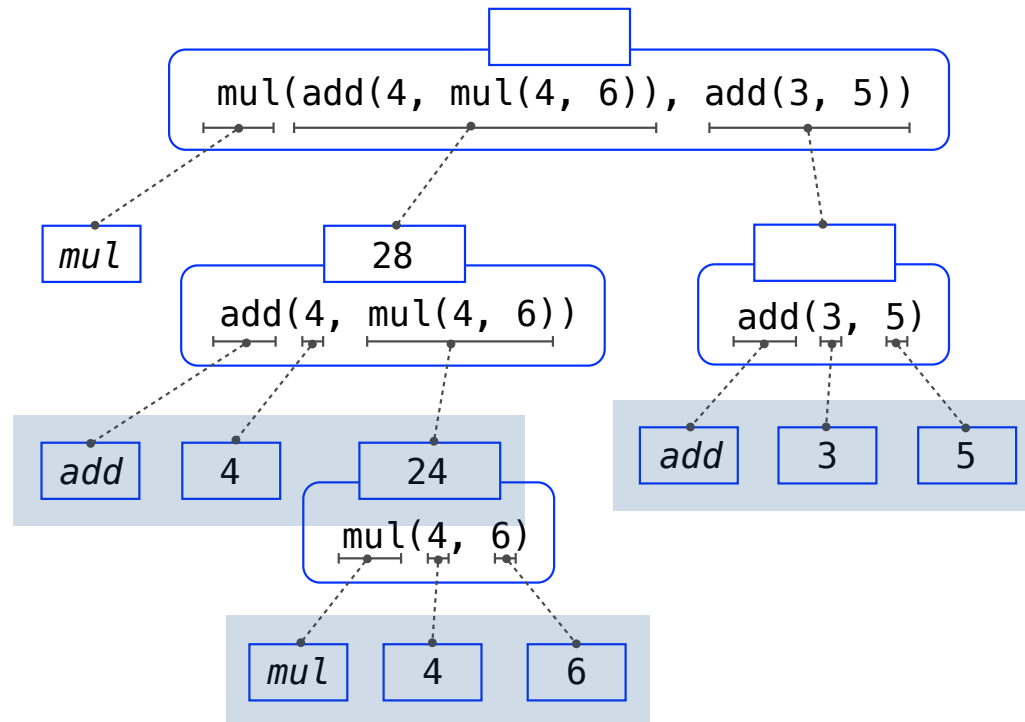
Evaluating Nested Expressions



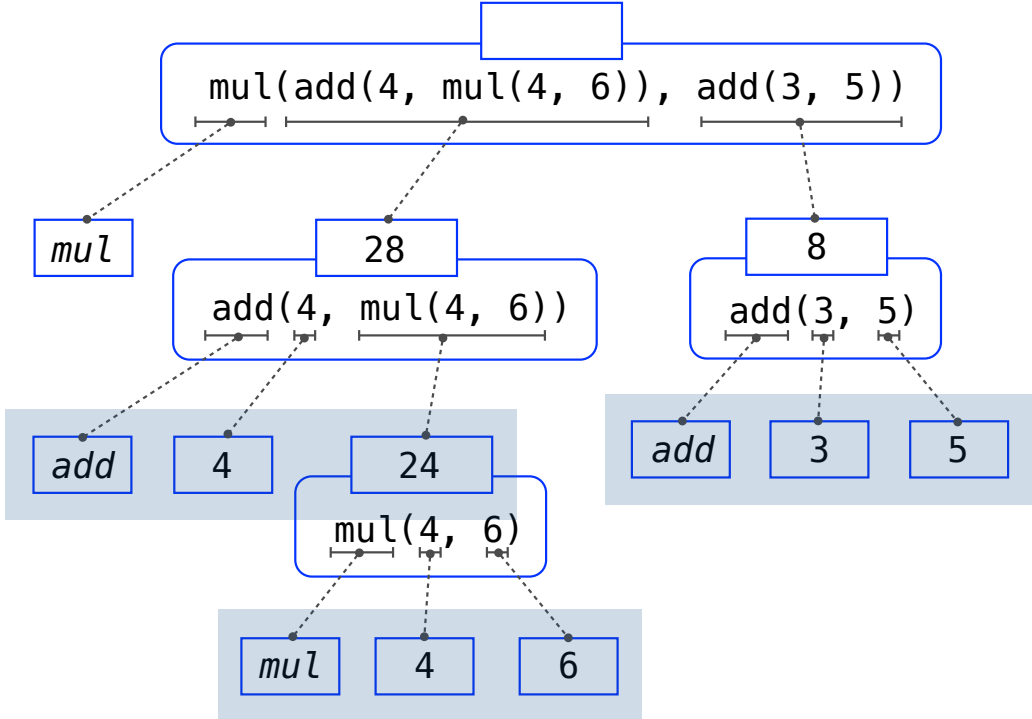
Evaluating Nested Expressions



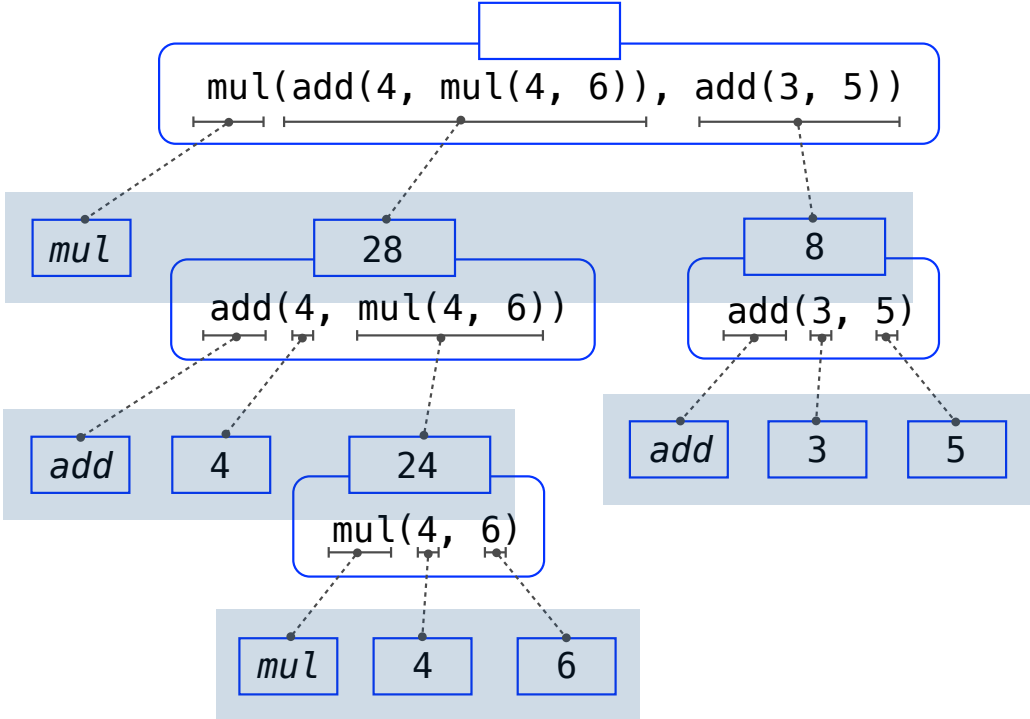
Evaluating Nested Expressions



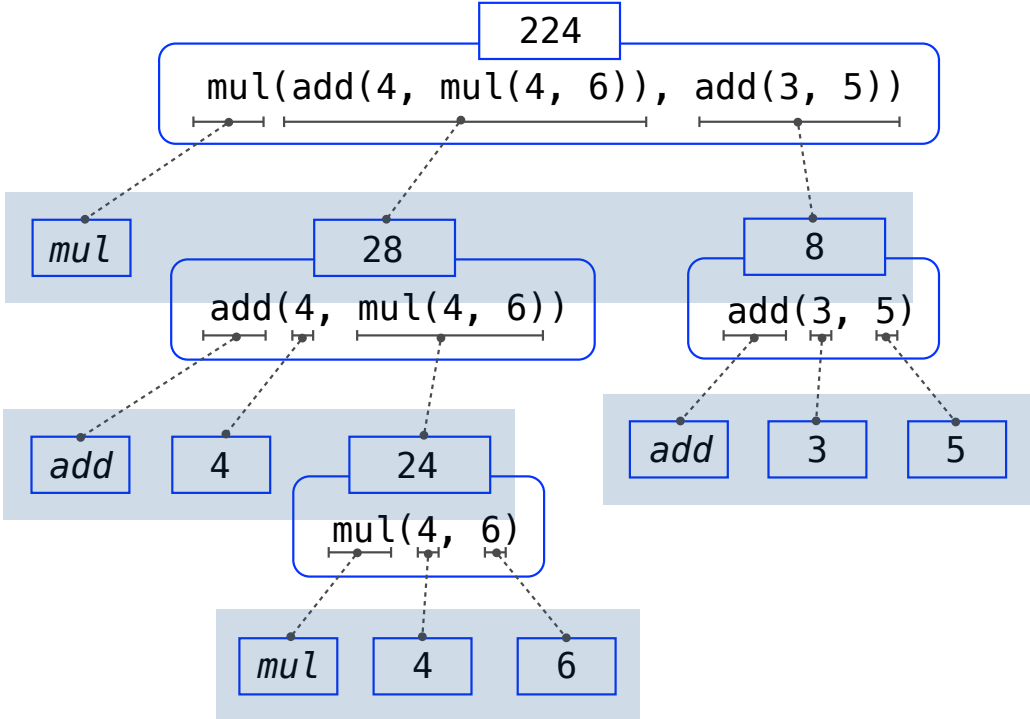
Evaluating Nested Expressions



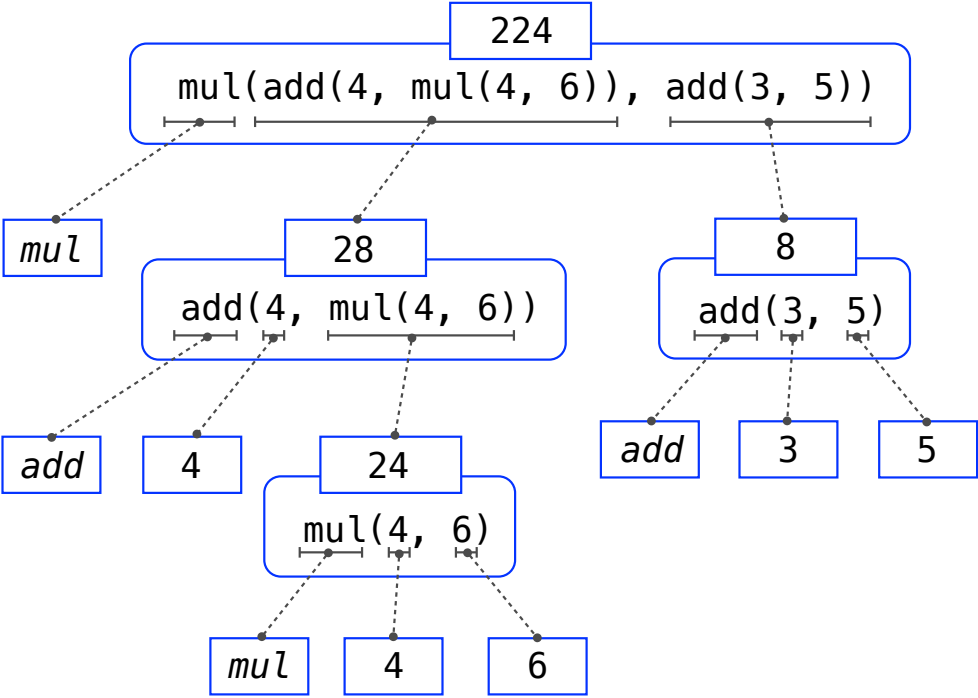
Evaluating Nested Expressions



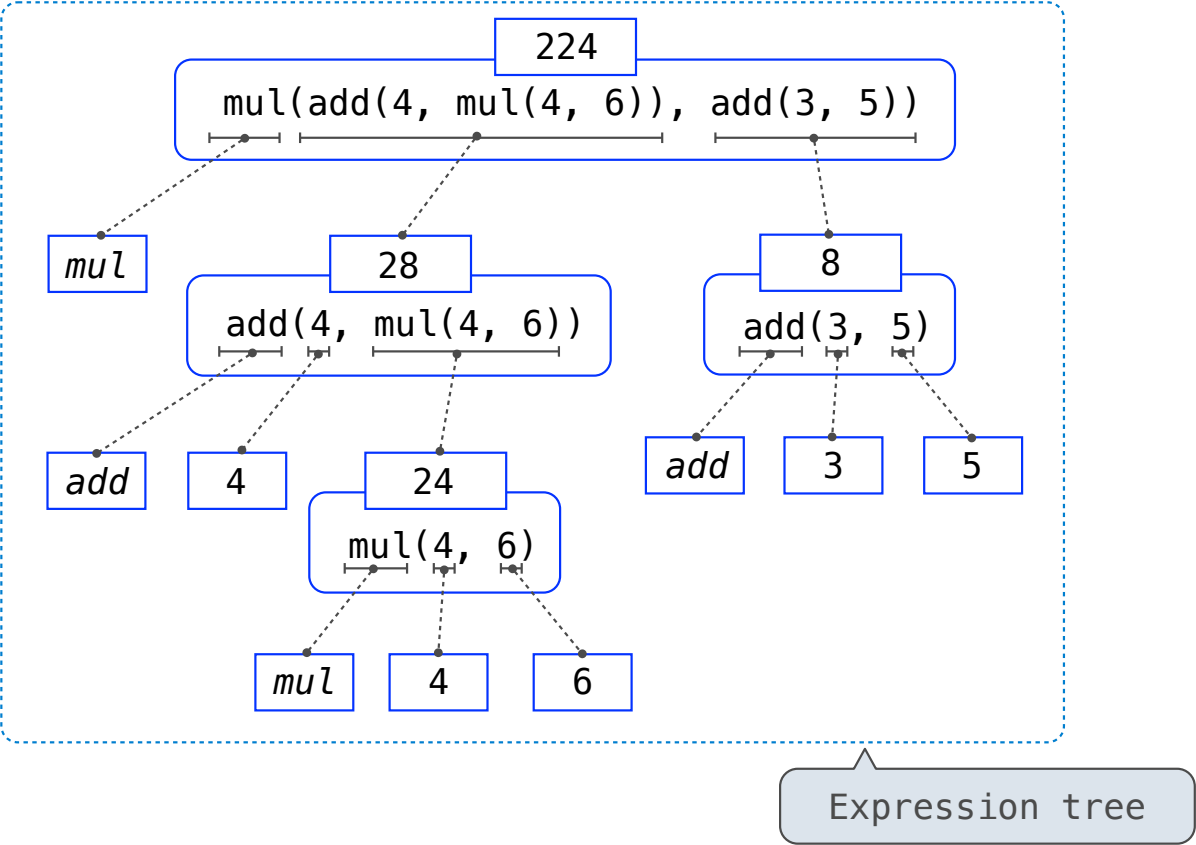
Evaluating Nested Expressions



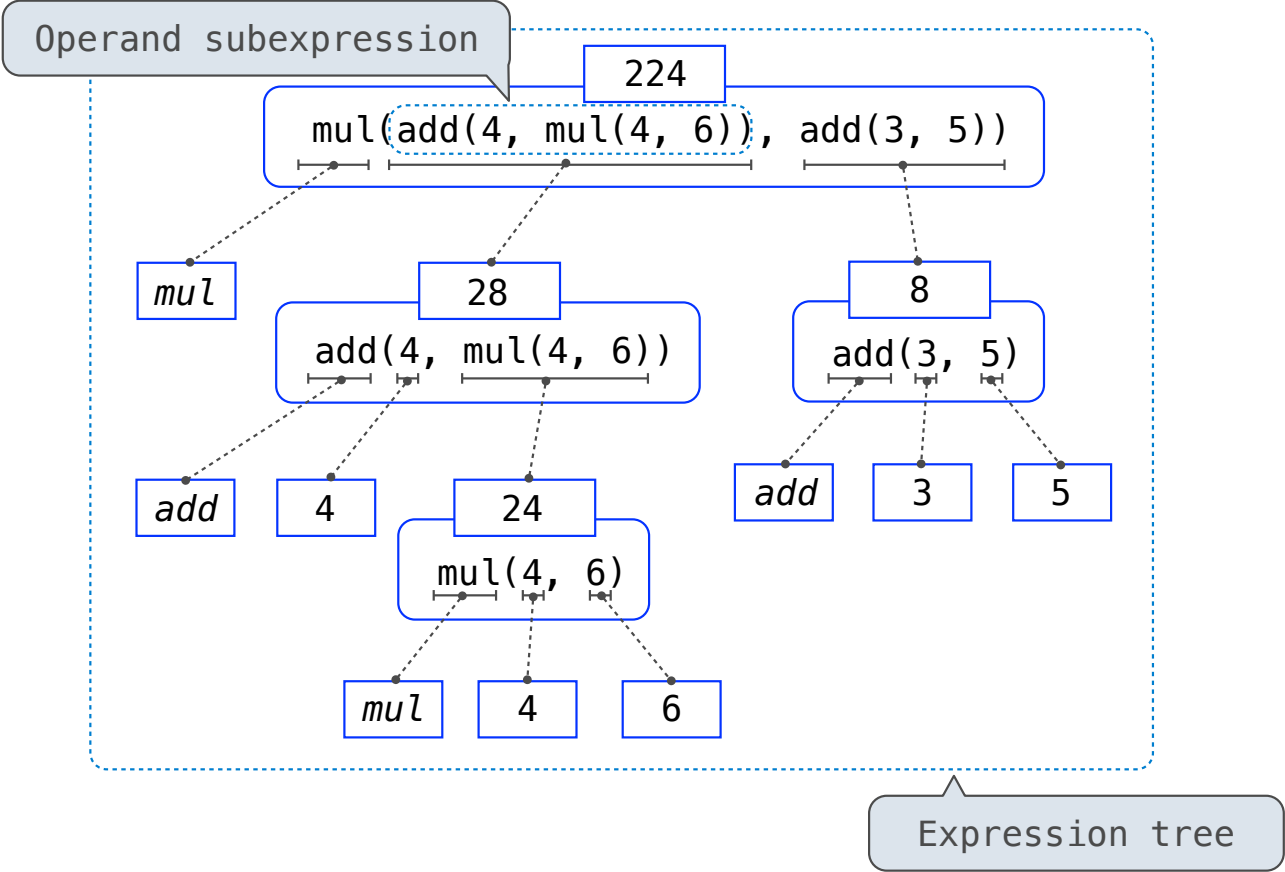
Evaluating Nested Expressions



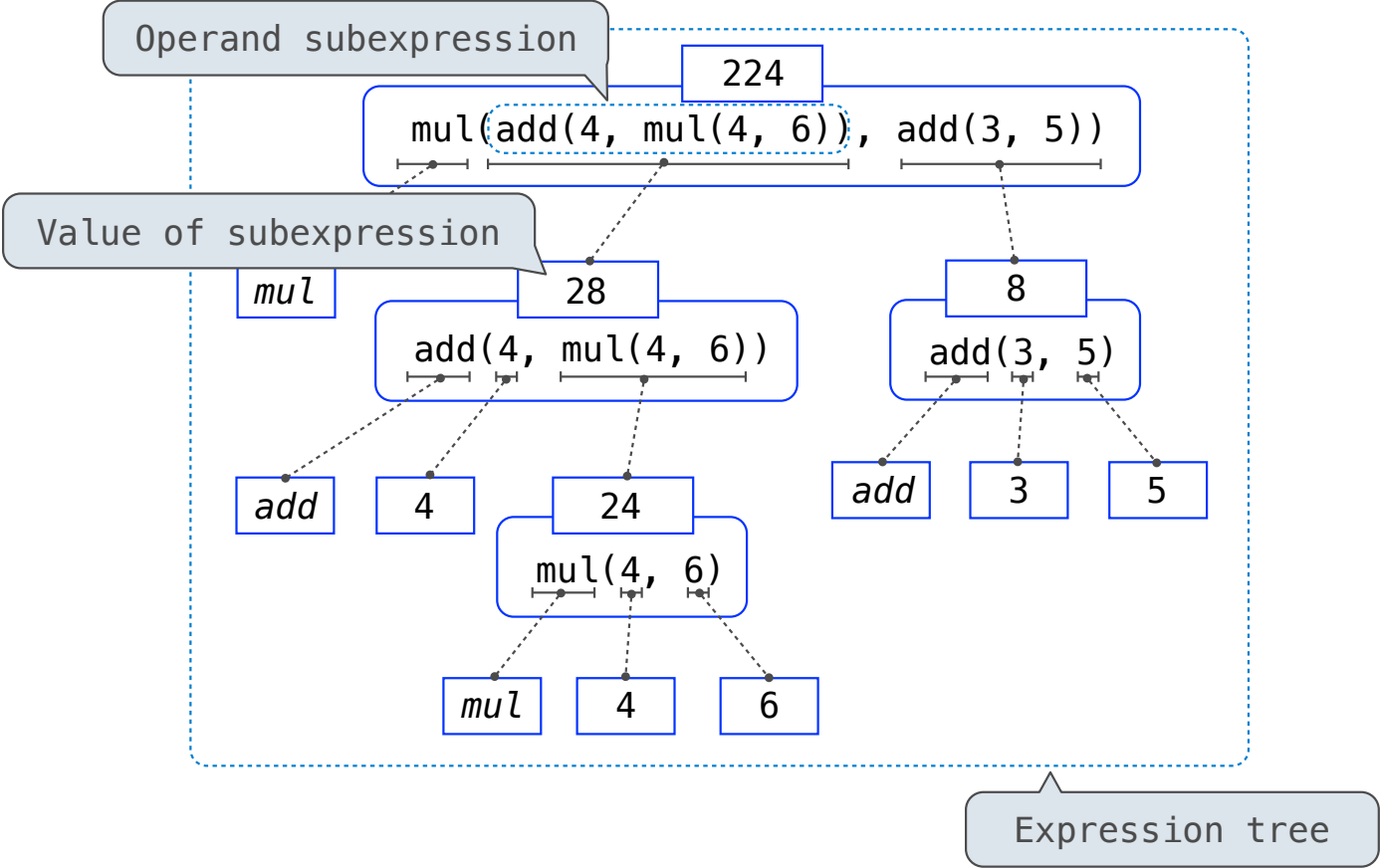
Evaluating Nested Expressions



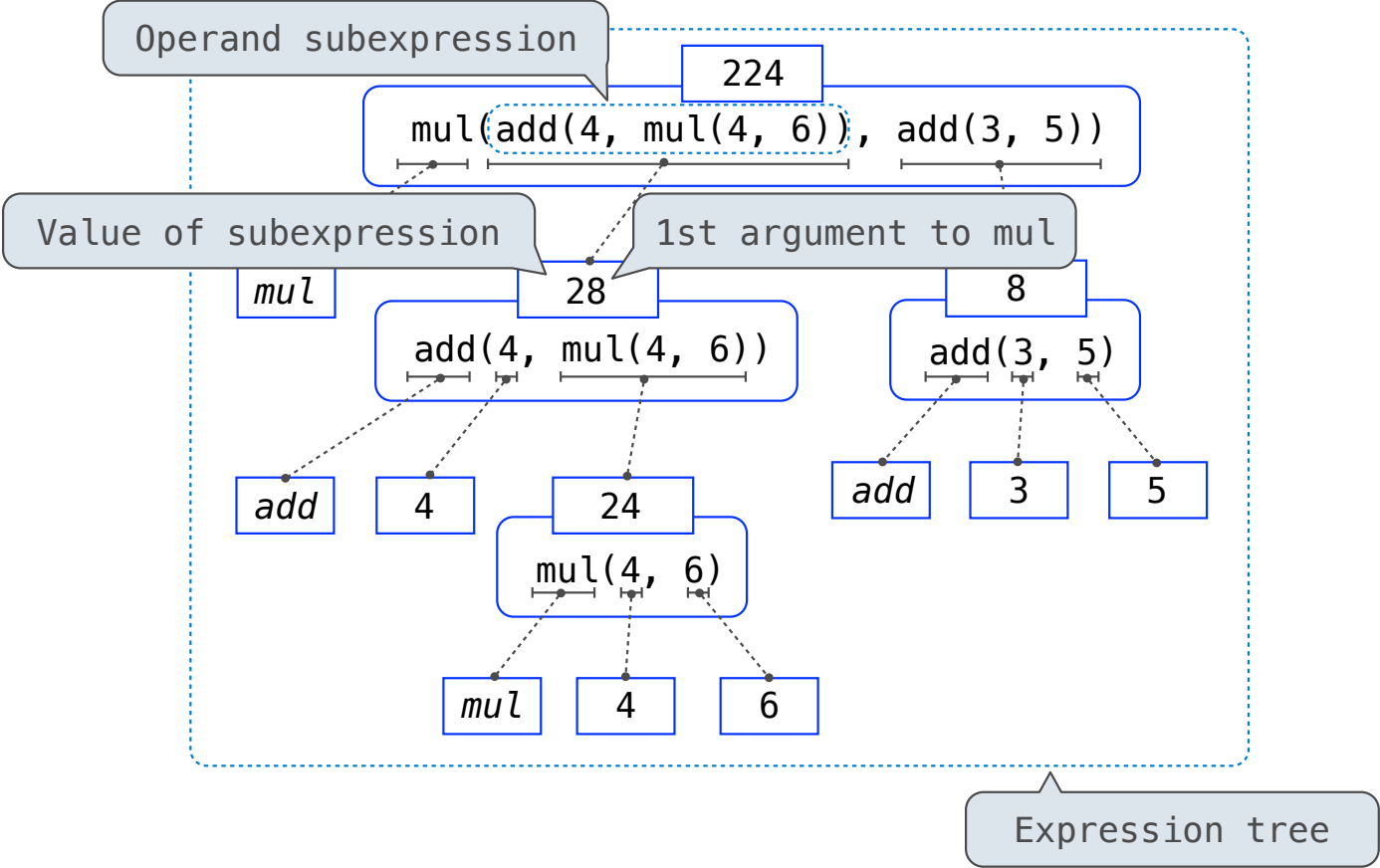
Evaluating Nested Expressions



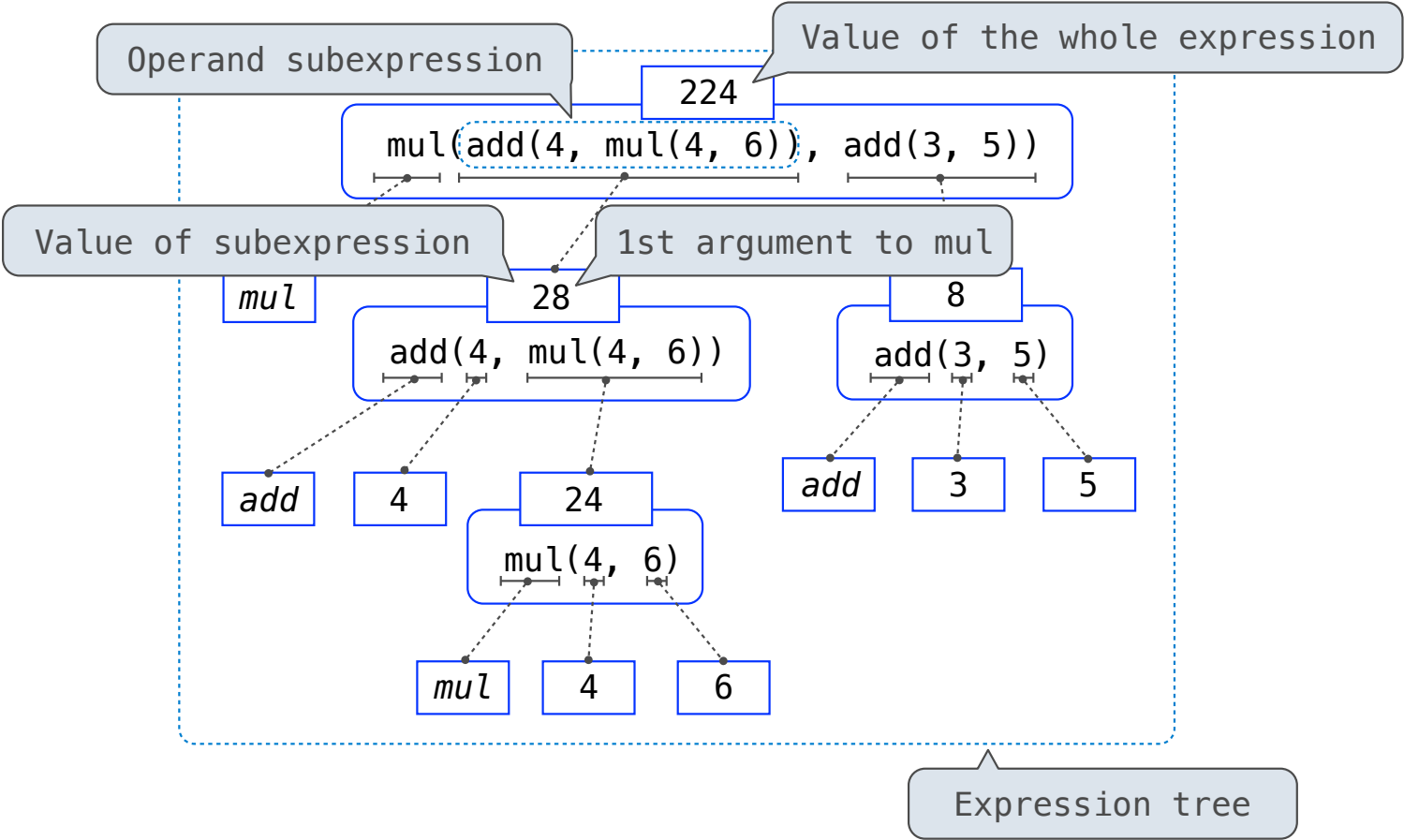
Evaluating Nested Expressions



Evaluating Nested Expressions



Evaluating Nested Expressions



Functions, Values, Objects, Interpreters, and Data

(Demo)