Welcome to Berkeley Computer Science!
Welcome to Berkeley Computer Science!

Soda Hall
Welcome to Berkeley Computer Science!
Welcome to Berkeley Computer Science!

How to contact me:
Welcome to Berkeley Computer Science!

How to contact me:

denero@berkeley.edu
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:
781 Soda
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:
781 Soda
Monday 3pm–4pm
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:
781 Soda
Monday 3pm–4pm
Thursday 10am–11am
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:
781 Soda
Monday 3pm–4pm
Thursday 10am–11am
Welcome to Berkeley Computer Science!

How to contact me:
denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:

**781 Soda**

Monday 3pm–4pm

Thursday 10am–11am

Fridays by appointment
Welcome to Berkeley Computer Science!

How to contact me:

denero@berkeley.edu
piazza.com/berkeley/fall2015/cs61a

Fall 2015 office hours:

**781 Soda**

Monday 3pm–4pm

Thursday 10am–11am

Fridays by appointment
denero.org/meet
The Course Staff
The Course Staff

40+ Teaching Assistants (GSIs/UGSIs) run labs, discussions, and office hours
The Course Staff

40+ Teaching Assistants (GSIs/UGSIs) run labs, discussions, and office hours
30+ Tutors are your personal programming mentors
The Course Staff

40+ **Teaching Assistants** (GSIs/UGSIs) run labs, discussions, and office hours

30+ **Tutors** are your personal programming mentors

150+ **Lab Assistants** ensure that you don’t get stuck for too long
The Course Staff

40+ **Teaching Assistants** (GSIs/UGSIs) run labs, discussions, and office hours

30+ **Tutors** are your personal programming mentors

150+ **Lab Assistants** ensure that you don’t get stuck for too long
**The Course Staff**

- **40+ Teaching Assistants** (GSIs/UGSIs) run labs, discussions, and office hours
- **30+ Tutors** are your personal programming mentors
- **150+ Lab Assistants** ensure that you don’t get stuck for too long
The Course Staff

40+ Teaching Assistants (GSIs/UGSIs) run labs, discussions, and office hours

30+ Tutors are your personal programming mentors

150+ Lab Assistants ensure that you don’t get stuck for too long
The Course Staff

40+ **Teaching Assistants** (GSIs/UGSIs) run labs, discussions, and office hours

30+ **Tutors** are your personal programming mentors

150+ **Lab Assistants** ensure that you don’t get stuck for too long
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:** The most important events in this course
Parts of the Course

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

**Lab:** The most important events in this course

**Discussion:** Also the most important events in this course
Parts of the Course

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

**Lab:** The most important events in this course

**Discussion:** Also the most important events in this course

**Office Hours:** Also the most important events in this course [11–5 M–Th & 11–1 Friday]
Parts of the Course

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

**Lab:** The most important events in this course

**Discussion:** Also the most important events in this course

**Office Hours:** Also the most important events in this course [11–5 M–Th & 11–1 Friday]

**Online textbook:** http://composingprograms.com
Parts of the Course

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

**Lab:** The most important events in this course

**Discussion:** Also the most important events in this course

**Office Hours:** Also the most important events in this course [11–5 M–Th & 11–1 Friday]

**Online textbook:** http://composingprograms.com

Weekly homework assignments, three exams, quizzes, & four programming projects
Parts of the Course

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

Lab: The most important events in this course

Discussion: Also the most important events in this course

Office Hours: Also the most important events in this course [11–5 M–Th & 11–1 Friday]

Online textbook: http://composingprograms.com

Weekly homework assignments, three exams, quizzes, & four programming projects

Lots of special events
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?

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

The study of Systems
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?

- Systems
- Artificial Intelligence
- Graphics
- Security
- Networking
- Programming Languages
- Theory
- Scientific Computing
- The study of what problems can be solved using computation,
- How to solve those problems, and
- What techniques lead to effective solutions
- Decision Making
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

...
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
Natural Language Processing
Networking
Programming Languages
Programing 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
Translation

...
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

Natural Language Processing

Networking

Translation

Programming Languages

Answering Questions

Theory

Scientific Computing

...
What is Computer Science?

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

The study of

Systems
Artificial Intelligence
Graphics
Security
Networking
Programming Languages
Theory
Scientific Computing

... Decision Making

Robotics
Natural Language Processing

... Translation

Answering Questions

...
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

Translation
Answering Questions
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
  - Using programming paradigms
What is This Course About?

• A course about managing complexity
  • Mastering abstraction
  • Using programming paradigms
  • Completing big projects
What is This Course About?

- A course about managing complexity
  - Mastering abstraction
  - Using programming paradigms
  - Completing big projects
What is This Course About?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Using programming paradigms
  ▪ Completing big projects

• An introduction to Python
What is This Course About?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Using programming paradigms
  ▪ Completing big projects

• An introduction to Python
  ▪ Full understanding of fundamentals
What is This Course About?

- A course about managing complexity
  - Mastering abstraction
  - Using programming paradigms
  - Completing big projects
- An introduction to Python
  - Full understanding of fundamentals
  - Learning through implementation
What is This Course About?

• A course about managing complexity
  § Mastering abstraction
  § Using programming paradigms
  § Completing big projects
• An introduction to Python
  § Full understanding of fundamentals
  § Learning through implementation
  § How computers interpret programming languages
What is This Course About?

• A course about managing complexity
  § Mastering abstraction
  § Using programming paradigms
  § Completing big projects

• An introduction to Python
  § Full understanding of fundamentals
  § Learning through implementation
  § How computers interpret programming languages

• A challenging course that will demand a lot of you
Other Courses
CS 61AS: Self-Paced 61A

You choose the pace! The course can be completed over two semesters.
You choose the pace! The course can be completed over two semesters.

Extra content for people without prior programming experience.
CS 61AS: Self-Paced 61A

You choose the pace! The course can be completed over two semesters

Extra content for people without prior programming experience

A brilliant textbook, interesting projects, a great language, and a dedicated course staff
CS 61AS: Self-Paced 61A

You choose the pace! The course can be completed over two semesters

Extra content for people without prior programming experience

A brilliant textbook, interesting projects, a great language, and a dedicated course staff

More info: cs61as.org
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

More info: cs10.org
Data Science 8: Foundations of Data Science

Fundamentals of computing and inference applied to real-world data
Data Science 8: Foundations of Data Science

Fundamentals of computing and inference applied to real-world data
Data Science 8: Foundations of Data Science

Fundamentals of computing and inference applied to real-world data

Great programming practice for CS 61A
Data Science 8: Foundations of Data Science

Fundamentals of computing and inference applied to real-world data

Great programming practice for CS 61A

In Fall 2015, piloted as Stat 94 (CCN: 87470)
Data Science 8: Foundations of Data Science

Fundamentals of computing and inference applied to real-world data

Great programming practice for CS 61A

In Fall 2015, piloted as Stat 94 (CCN: 87470)

More info: data8.org & databears.berkeley.edu
Course Policies
Course Policies

Learning
Course Policies

Learning

Community
Course Policies

Learning
Community
Course Staff
Course Policies

Learning

Community

Course Staff

Details...

http://cs61a.org/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!
- Homework can be completed with a partner
Collaboration

Asking questions is highly encouraged

- Discuss everything with each other; learn from your fellow students!
- Homework can be completed with a partner
- Projects should be completed with a partner
**Collaboration**

**Asking questions is highly encouraged**

- Discuss everything with each other; learn from your fellow students!
- Homework can be completed with a partner
- Projects should 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!
• Homework can be completed with a partner
• Projects should 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!
- Homework can be completed with a partner
- Projects should 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 partner
Collaboration

**Asking questions is highly encouraged**
- Discuss everything with each other; learn from your fellow students!
- Homework can be completed with a partner
- Projects should 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 partner
- Copying project solutions causes people to fail this course
Collaboration

**Asking questions is highly encouraged**
- Discuss everything with each other; learn from your fellow students!
- Homework can be completed with a partner
- Projects should 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 partner
- Copying project solutions causes people to fail this course
- We really do catch people who violate the rules, because...
Collaboration

Asking questions is highly encouraged
• Discuss everything with each other; learn from your fellow students!
• Homework can be completed with a partner
• Projects should 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 partner
• Copying project solutions causes people to fail this course
• We really do catch people who violate the rules, because...
  • We also know how to search the web for solutions
**Collaboration**

**Asking questions is highly encouraged**
- Discuss everything with each other; learn from your fellow students!
- Homework can be completed with a partner
- Projects should 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 partner
- Copying project solutions causes people to fail this 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
Collaboration

Asking questions is highly encouraged
• Discuss everything with each other; learn from your fellow students!
• Homework can be completed with a partner
• Projects should 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 partner
• Copying project solutions causes people to fail this 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
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

\[ 18 + 69 = 87 \]

\[ \frac{6}{23} \]

\[ \sqrt{3493161} \]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \quad 6 \quad \sin \pi \quad \frac{6}{23} \quad \sqrt{3493161} \]
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

\[ 18 + 69 \]
\[ \frac{6}{23} \]
\[ \sin \pi \]
\[ f(x) \]
\[ \sqrt{3493161} \]
\[ \sum_{i=1}^{100} i \]
\[ | -1869| \]
\[ (69) \]
\[ (18) \]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \]
\[ \frac{6}{23} \]
\[ \sin \pi \]
\[ 2^{100} \]
\[ f(x) \]
\[ \sqrt{3493161} \]
\[ \sum_{i=1}^{100} i \]
\[ | - 1869| \]
\[ \binom{69}{18} \]
Types of expressions

An expression describes a computation and evaluates to a value

\[
\begin{align*}
18 + 69 & \quad \frac{6}{23} & \quad \sin \pi & \quad \log_2 1024 \\
2^{100} & \quad f(x) & \quad \sqrt{3493161} & \\
| - 1869| & \quad \sum_{i=1}^{100} i & \quad (69) & \quad (18)
\end{align*}
\]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \quad \frac{6}{23} \quad \sin \pi \quad \log_2 1024 \]

\[ 2^{100} \quad f(x) \quad \sqrt{3493161} \]

\[ 7 \mod 2 \quad \sum_{i=1}^{100} i \quad \left( \begin{array}{c} 69 \\ 18 \end{array} \right) \]

\[ | -1869| \]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \]
\[ \frac{6}{23} \]
\[ \sin \pi \]
\[ \log_2 1024 \]
\[ 2^{100} \]
\[ f(x) \]
\[ 7 \mod 2 \]
\[ | - 1869| \]
\[ \sum_{i=1}^{100} i \]
\[ \sqrt{3493161} \]
\[ \left( \begin{array}{c} 69 \\ 18 \end{array} \right) \]
\[ \lim_{x \to \infty} \frac{1}{x} \]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \]
\[ \frac{6}{23} \]
\[ \sin \pi \]
\[ \log_2 1024 \]
\[ 2^{100} \]
\[ f(x) \]
\[ 7 \mod 2 \]
\[ | - 1869| \]
\[ \sum_{i=1}^{100} i \]
\[ \sqrt{3493161} \]
\[ \lim_{x \to \infty} \frac{1}{x} \]
\[ \left( \begin{array}{c} 69 \\ 18 \end{array} \right) \]
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

\[
\text{add} \quad ( \quad 2 \quad , \quad 3 \quad )
\]

Operator
Anatomy of a Call Expression

\[
\text{add} \ ( \ 2 \ , \ 3 \ )
\]

Operator \hspace{1cm} Operand \hspace{1cm} Operand
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

Evaluation procedure for call expressions:
Anatomy of a Call Expression

Evaluation procedure for call expressions:

1. Evaluate the operator and then the operand subexpressions
Anatomy of a Call Expression

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

Operators and operands are also expressions

So they evaluate to values
Evaluating Nested Expressions

\texttt{mul(add(2, mul(4, 6)), add(3, 5))}
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

mul(add(2, mul(4, 6)), add(3, 5))
Evaluating Nested Expressions
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

mul(add(2, mul(4, 6)), add(3, 5))
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

mul(add(2, mul(4, 6)), add(3, 5))

mul

add(2, mul(4, 6))

add

2

mul

4

6

mul

24
Evaluating Nested Expressions

mul(add(2, mul(4, 6)), add(3, 5))

[Diagram of nested expressions]
Evaluating Nested Expressions

\[ \text{mul}(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)) \]
Evaluating Nested Expressions

\[
mul(add(2, mul(4, 6)), add(3, 5))
\]
Evaluating Nested Expressions

mul(add(2, mul(4, 6)), add(3, 5))
Evaluating Nested Expressions

\[ \text{mul}\left(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5)\right) \]
Evaluating Nested Expressions

Expression tree
Evaluating Nested Expressions

```
mul(add(2, mul(4, 6)), add(3, 5))
```

Expression tree

Operand subexpression
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

208
mul(add(2, mul(4, 6)), add(3, 5))

26
mul
add(2, mul(4, 6))

8
add(3, 5)

24
mul(4, 6)

2
add
mul
4
6

Expression tree
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

1st argument to mul

mul(\text{add}(2, \text{mul}(4, 6)), \text{add}(3, 5))

mul

26

add(2, \text{mul}(4, 6))

add

2

24

\text{mul}(4, 6)

mul

4

6

add

3

5

8

add(3, 5)

Expression tree
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

1st argument to mul

Value of the whole expression

208

mul(add(2, mul(4, 6)), add(3, 5))

mul

26

add(2, mul(4, 6))

add

2

mul

24

mul(4, 6)

mul

4

6

8

add(3, 5)

add

3

5

Expression tree
Functions, Objects, and Interpreters

(Demo)