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

Fall 2014 office hours:

411 Soda
Tuesday 12pm–1pm
Wednesday 12pm–1pm

781 Soda by appointment
http://denero.org/meet
The Course Staff
Teaching Assistants (UGSIs/GSIs) run discussion sections, labs, and office hours.
The Course Staff

Teaching Assistants (UGSIs/GSIs) run discussion sections, labs, and office hours.

Soumya Basu  Matthew Chow  Ajeya Cotra  Brian Hou  Andrew Huang  Robert Huang  Michelle Hwang  Mehdi Jaremi
Joy Jong  Chloe Lischinsky  Kaylee Mann  Beth Marrone  Allen Nguyen  Youri Park  Jack Qiao  Sumukh Sridhara
Steven Tang  Michael Tao  Dickson Tsai  Iris Wang  Albert Wu  Chenyang Yuan  Marvin Zhang
The Course Staff

Teaching Assistants (UGSIs/GSIs) run discussion sections, labs, and office hours.

18 Readers are your personal programming mentors.
### The Course Staff

**Teaching Assistants (UGSIs/GSIs)** run discussion sections, labs, and office hours.

<table>
<thead>
<tr>
<th>Soumya Basu</th>
<th>Matthew Chow</th>
<th>Ajaya Cotra</th>
<th>Brian Hou</th>
<th>Andrew Huang</th>
<th>Robert Huang</th>
<th>Michelle Hwang</th>
<th>Mehdi Jarraei</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy Jong</td>
<td>Chloe Lischnisky</td>
<td>Kaylee Mann</td>
<td>Beth Marrone</td>
<td>Allen Nguyen</td>
<td>Youri Park</td>
<td>Jack Qiao</td>
<td>Sumukh Sridhara</td>
</tr>
<tr>
<td>Steven Tang</td>
<td>Michael Tao</td>
<td>Dickson Tsai</td>
<td>Iris Wang</td>
<td>Albert Wu</td>
<td>Chenyang Yuan</td>
<td>Marvin Zhang</td>
<td></td>
</tr>
</tbody>
</table>

18 **Readers** are your personal programming mentors.

Over 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 http://cs61a.org before each live lecture
Parts of the Course

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

**Lab:** The most important events in this course
Parts of the Course

**Lecture:** Videos posted to [http://cs61a.org](http://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 [http://cs61a.org](http://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 every day in 411 Soda]
Parts of the Course

**Lecture:** Videos posted to [http://cs61a.org](http://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 every day in 411 Soda]

**Online textbook:** [http://composingprograms.com](http://composingprograms.com)
Parts of the Course

**Lecture:** Videos posted to [http://cs61a.org](http://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 every day in 411 Soda]

**Online textbook:** [http://composingprograms.com](http://composingprograms.com)

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

Lecture: Videos posted to http://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 every day in 411 Soda]

Online textbook: http://composingprograms.com

Weekly homework assignments, three exams, & four programming projects

Many 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?

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

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

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?

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
...
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

...
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?

- 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
- Decision Making
- Robotics
- Natural Language Processing
- 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
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 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 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
- Not all about 0's and 1's
What is This Course About?

• A course about managing complexity

• Mastering abstraction

• Programming paradigms

• Not all about 0's and 1's
What is This Course About?

- A course about managing complexity
- Mastering abstraction
- Programming paradigms
- Not all about 0's and 1's
- An introduction to Python
What is This Course About?

• A course about managing complexity
  • Mastering abstraction
  • Programming paradigms
  • Not all about 0's and 1's

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

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Programming paradigms
  ▪ Not all about 0's and 1's

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

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Programming paradigms
  ▪ Not all about 0's and 1's

• An introduction to Python
  ▪ Full understanding of language fundamentals
  ▪ Learning through implementation
  ▪ How computers interpret programming languages
What is This Course About?

• A course about managing complexity
  • Mastering abstraction
  • Programming paradigms
  • Not all about 0's and 1's

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

• A challenging course that will demand a lot of you
Course Policies
Alternatives to This Course
Alternatives to This Course

CS 61AS: Self-Paced 61A
Alternatives to This Course

CS 61AS: Self-Paced 61A

CS 10: The Beauty and Joy of Computing
Course Policies
Learning
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 know how to use computers
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 know how to use computers

**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

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

An expression describes a computation and evaluates to a value

\[ 18 + 69 \quad \frac{6}{23} \quad \sin \pi \quad \sqrt{3493161} \quad | - 1869| \]
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.

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

An expression describes a computation and evaluates to a value

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

\[
f(x) \quad \sum_{i=1}^{100} i \quad \sqrt{3493161} \\
| - 1869| \\
(69) \quad (18)
\]
Types of expressions

An expression describes a computation and evaluates to a value

$18 + 69$

$\frac{6}{23}$

$\sin \pi$

$2^{100}$

$f(x)$

$\sqrt{3493161}$

$\left| -1869 \right|$

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

$\binom{69}{18}$
Types of expressions

An expression describes a computation and evaluates to a value

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

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

\[ \sum_{i=1}^{100} i \quad | - 1869| \quad \binom{69}{18} \]
Types of expressions

An expression describes a computation and evaluates to a value

\[ 18 + 69 \]

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

\[ \sin \pi \]

\[ \log_2 1024 \]

\[ 2^{100} \]

\[ f(x) \]

\[ 7 \mod 2 \]

\[ | - 1869 | \]

\[ \sum_{i=1}^{100} i \]

\[ \sqrt{3493161} \]

\[ (69) \]

\[ (18) \]
Types of expressions

An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

Operator
Anatomy of a Call Expression

```
add ( 2 , 3 )
```

Operator
Operand
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

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 subexpression to the **arguments** that are the values of the operand subexpression
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

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

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

Diagram:
- \text{mul}
- \text{add}
- \text{mul}(2, \text{mul}(4, 6))
Evaluating Nested Expressions
Evaluating Nested Expressions

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

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

```
mul

mul(4, 6)
mul

add(2, mul(4, 6))

add

2

mul(4, 6)

mul

4

6
```
Evaluating Nested Expressions

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

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

- \text{mul}(\text{add}(2, 24), 26)
- \text{add}(2, 26)
- \text{mul}(4, 6)
- \text{mul}(2, 24)
- \text{add}(2, 26)
- \text{mul}(4, 6)
- \text{mul}(2, 24)
- \text{add}(2, 26)
- \text{mul}(4, 6)
Evaluating Nested Expressions
Evaluating Nested Expressions
Evaluating Nested Expressions

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

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

```
208
mul
26
add(2, mul(4, 6))
add
2
24
mul(4, 6)
mul
4
6
8
add(3, 5)
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

\[ \text{Operand subexpression} \]

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

\[ \text{Expression tree} \]
Evaluating Nested Expressions

Expression tree

Operand subexpression

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

1st argument to mul

Expression tree
Evaluating Nested Expressions

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

1st argument to mul

Value of the whole expression

Operand subexpression

Expression tree
Evaluating Nested Expressions

Operand subexpression

Value of the whole expression

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

1st argument to mul

Expression tree
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))
```

- **Operand subexpression**: `mul(add(2, mul(4, 6)), add(3, 5))`
- **1st argument to mul**: `add(2, mul(4, 6))`
- **Value of the whole expression**: `add(2, mul(4, 6))`

**Expression tree**
Evaluating Nested Expressions

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

Expression tree
Operand subexpression
Value of the whole expression
1st argument to mul
Expression tree
Evaluating Nested Expressions

Operand subexpression

Value of the whole expression

Expression tree

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

1st argument to mul

add(2, mul(4, 6))

add

2
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of the whole expression

1st argument to mul

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

mul

add(2, mul(4, 6))

add 2

mul(4, 6)

Expression tree
Evaluating Nested Expressions

Expression tree:

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

Operand subexpression:

```
mul
```

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

```
add
```

```
mul
```

```
mul(4, 6)
```

1st argument to mul:

```
mul
```

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

Value of the whole expression:

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

Expression tree

Operand subexpression

Value of the whole expression

1st argument to \texttt{mul}

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

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

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

\texttt{mul}(4, 6)

\texttt{mul}(4, 6)

\texttt{mul}(4, 6)
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of the whole expression

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

mul

add(2, mul(4, 6))

add

2

mul(4, 6)

mul

4

6

1st argument to mul

26

Value of the whole expression

Evaluating Nested Expressions
Evaluating Nested Expressions
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of the whole expression

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

1st argument to mul

mul

add(2, mul(4, 6))

add

2

24

mul(4, 6)

mul

4

6

add

3

5

add(3, 5)
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of the whole expression

1st argument to mul

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

mul

26

add(2, mul(4, 6))

mul(4, 6)

mul

4

6

add

2

24

add(3, 5)

add

3

5

Expression tree
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of the whole expression

208

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

1st argument to mul

mul

26

add(2, mul(4, 6))

add

2

24

mul(4, 6)

mul

4

6

add

3

5

add(3, 5)

8

Expression tree
Evaluating Nested Expressions

Expression tree

Operand subexpression
Value of subexpression
Value of the whole expression

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

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

add
2
mul
4
6

208

8
add(3, 5)

add
3
5

add(2, mul(4, 6))
Functions, Objects, and Interpreters

(Demo)