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

Spring 2015 office hours:

781 Soda
Wednesday 10am–12pm &
Friday by appointment:
http://denero.org/meet
The Course Staff
The Course Staff

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

Teaching Assistants (GSIs/UGSIs) run discussion sections, labs, and office hours
The Course Staff

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

27 **Group Tutors** are your personal programming mentors
The Course Staff

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

27 **Group Tutors** are your personal programming mentors

Over 300 **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](http://cs61a.org) before each live lecture
Parts of the Course

**Lecture:** Videos posted to 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 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 M–Th & 11–1 Friday]
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 M–Th & 11–1 Friday]

**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 M–Th & 11–1 Friday]

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

Weekly homework assignments, three exams, three quizzes, & four programming projects
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 M–Th & 11–1 Friday]

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

Weekly homework assignments, three exams, three 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?

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?

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
What is Computer Science?

The study of systems, artificial intelligence, and graphics.

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
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 Computation includes:

- 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

Decision Making
Robotics
Natural Language Processing

...
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
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
Robotics
Natural Language Processing
Translation
Answering Questions

...
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
- Decision Making
- Graphics
- Robotics

**Security**
- Natural Language Processing
- Translation

**Networking**
- Programming Languages
- Answering Questions

**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
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
  • Programming paradigms
What is This Course About?

• A course about managing complexity
  • Mastering abstraction
  • Programming paradigms
  • Not just about 0's and 1's
What is This Course About?

• A course about managing complexity
• Mastering abstraction
• Programming paradigms
• Not just about 0's and 1's
What is This Course About?

• A course about managing complexity

  • Mastering abstraction

  • Programming paradigms

  • Not just 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 just 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 just 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 just 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 just 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 CS 61A
Alternatives to This Course

CS 61AS: Self-Paced CS 61A

CS 10: The Beauty and Joy of Computing
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 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
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

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

An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

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

\[ \sum_{i=1}^{100} i \quad \left| -1869 \right| \]

\[ (69) \quad (18) \]
Types of expressions

An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

\[ 2^{100} \]

\[ f(x) \]

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

\[ | - 1869| \]

\[ \sqrt{3493161} \]

\[ (69) \quad (18) \]
An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

An expression describes a computation and evaluates to a value

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

All expressions can use function call notation
(Demo)
Anatomy of a Call Expression
Anatomy of a Call Expression

\[ \text{add}(2, 3) \]
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: `add`
- Operands: `2`, `,`, `3`
Anatomy of a Call Expression

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

Operator \quad \text{Operand} \quad \text{Operand}

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:**

```
add ( 2, 3 )
Operator    Operand    Operand
```
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
Evaluating Nested Expressions

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

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

mul

26

add(2, mul(4, 6))

add

2

24

mul(4, 6)

mul

4

6

mul

8

add(3, 5)

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

Expression tree:

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

- `mul`: multiplication
- `add`: addition

Values:
- `2`
- `3`
- `4`
- `5`
- `6`
- `24`
- `26`
- `8`
- `208`
Evaluating Nested Expressions

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

Expression tree

Operand subexpression
Evaluating Nested Expressions
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

1st argument to mul

mul

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

208

mul

add(2, mul(4, 6))

26

add

2

mul

4

6

mul

4

6

add
d

3

5

add

add(3, 5)

8

add

3

5
Evaluating Nested Expressions

Expression tree

Operand subexpression

Value of subexpression

1st argument to mul

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

add

3

5

208

26

mul(4, 6)

24

8
Functions, Objects, and Interpreters

(Demo)