Welcome to Berkeley Computer Science!
Humans of CS 61A
Humans of CS 61A

2 Lecturers
Humans of CS 61A

2 Lecturers

12 TAs
Humans of CS 61A

2 Lecturers

12 TAs

13 Tutors
Humans of CS 61A

2 Lecturers

12 TAs

13 Tutors

100+ Lab assistants!
Humans of CS 61A

2 Lecturers

12 TAs

13 Tutors

100+ Lab assistants!

400+ Students!!!
Computer Science in one slide
Computer Science in one slide

• What problems can computers solve?
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence

Security
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking

Theory
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking

Theory

Computational Biology
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking

Theory

Computational Biology

...
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems
Artificial Intelligence
Security
Networking
Theory
Computational Biology
...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems
Artificial Intelligence
Security
Networking
Theory
Computational Biology

Natural Language Processing
Machine Learning
Computer Vision

...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems
Artificial Intelligence
Security
Networking
Theory
Computational Biology

...
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems
Artificial Intelligence
Security
Networking
Theory
Computational Biology

...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems  
Artificial Intelligence  
Security  
Networking  
Theory  
Computational Biology

Natural Language Processing  
Machine Learning  
Computer Vision  
Planning  
Robotics  
...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

- Systems
- Artificial Intelligence
- Security
- Networking
- Theory
- Computational Biology
- Natural Language Processing
- Machine Learning
- Computer Vision
- Planning
- Robotics
- ...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking

Theory

Computational Biology

...
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

Systems

Artificial Intelligence

Security

Networking

Theory

Computational Biology

Natural Language Processing

Machine Learning

Computer Vision

Planning

Robotics

Manipulation

Navigation and Locomotion

...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems
- Artificial Intelligence
- Security
- Networking
- Theory
- Computational Biology

- Natural Language Processing
- Machine Learning
- Computer Vision
- Planning
- Robotics
- ...
Computer Science in one slide

• What problems can computers solve?
• How do we get computers to solve these problems?
• What are general techniques for problem solving?

- Systems
- Artificial Intelligence
- Security
- Networking
- Theory
- Computational Biology
- Natural Language Processing
- Machine Learning
- Computer Vision
- Planning
- Robotics
- ...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?

Systems
Artificial Intelligence
Security
Networking
Theory
Computational Biology
...
Computer Science in one slide

- What problems can computers solve?
- How do we get computers to solve these problems?
- What are general techniques for problem solving?
CS 61A in one slide
CS 61A in one slide

• High-level ideas in computer science:
CS 61A in one slide

• High-level ideas in computer science:
  • *Abstraction*: manage complexity by hiding the details
• High-level ideas in computer science:
  • *Abstraction*: manage complexity by hiding the details
  • *Paradigms*: utilize different approaches to programming
High-level ideas in computer science:

- **Abstraction**: manage complexity by hiding the details
- **Paradigms**: utilize different approaches to programming
High-level ideas in computer science:

- **Abstraction**: manage complexity by hiding the details
- **Paradigms**: utilize different approaches to programming
CS 61A in one slide

• High-level ideas in computer science:
  • *Abstraction*: manage complexity by hiding the details
  • *Paradigms*: utilize different approaches to programming

• Master these ideas through implementation:
CS 61A in one slide

• High-level ideas in computer science:
  • *Abstraction*: manage complexity by hiding the details
  • *Paradigms*: utilize different approaches to programming

• Master these ideas through implementation:
  • Learn the Python programming language (& others)
CS 61A in one slide

• High-level ideas in computer science:
  • *Abstraction*: manage complexity by hiding the details
  • *Paradigms*: utilize different approaches to programming

• Master these ideas through implementation:
  • Learn the Python programming language (& others)
  • Complete large programming assignments
CS 61A in one slide

• High-level ideas in computer science:
  • Abstraction: manage complexity by hiding the details
  • Paradigms: utilize different approaches to programming

• Master these ideas through implementation:
  • Learn the Python programming language (& others)
  • Complete large programming assignments

• A challenging course that will demand a lot from you
Alternatives to CS 61A
Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing

[cs10.org](cs10.org) Offered this summer!
Alternatives to CS 61A

CS 10: The Beauty and Joy of Computing

[cs10.org]

Offered this summer!

Data Science 8: Foundations of Data Science

[data8.org]
Course Policies

Details on cs61a.org
Course overview
Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
Course overview

• Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
• Labs: the most important part of this course
Course overview

• Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
• Labs: the most important part of this course
• Discussions: the most important part of this course
Course overview

• Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
• Labs: the most important part of this course
• Discussions: the most important part of this course
• Office hours: the most important part of this course
Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com
Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com

- Regular homework assignments
Course overview

• Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
• Labs: the most important part of this course
• Discussions: the most important part of this course
• Office hours: the most important part of this course
• Online textbook: composingprograms.com

• Regular homework assignments
• 4 big programming projects
Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com

- Regular homework assignments
- 4 big programming projects
- Weekly quizzes, one midterm, and one final exam
Course overview

- Lectures: Mon–Thurs, 11am–12:30pm, 2050 VLSB
- Labs: the most important part of this course
- Discussions: the most important part of this course
- Office hours: the most important part of this course
- Online textbook: composingprograms.com

- Regular homework assignments
- 4 big programming projects
- Weekly quizzes, one midterm, and one final exam
- Lots of special events!
Grading
Grading
Grading

Assignments
150 points
Grading

Assignments 150 points

Exams 150 points
Grading

- Exams: 150 points
- Homework: 30 points
Grading

- Exams: 150 points
- Homework: 30 points
- Lab: 20 points
Grading

- Exams: 150 points
- Homework: 30 points
- Lab: 20 points
- Projects: 100 points
Grading

- Homework: 30 points
- Lab: 20 points
- Projects: 100 points
- Weekly quizzes: 40 points
Grading

- Homework: 30 points
- Lab: 20 points
- Projects: 100 points
- Weekly quizzes: 40 points
- Midterm: 40 points
Grading

- Homework: 30 points
- Lab: 20 points
- Projects: 100 points
- Weekly quizzes: 40 points
- Midterm: 40 points
- Final: 70 points
A few grading details
A few grading details

• 10 homework assignments, 3 points each
A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys
A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys

- 12 (graded) lab assignments, 2 points each
A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys

- 12 (graded) lab assignments, 2 points each
  - Two lowest lab scores will be dropped
A few grading details

• 10 homework assignments, 3 points each
  • Can make up points from one homework with surveys

• 12 (graded) lab assignments, 2 points each
  • Two lowest lab scores will be dropped

• Written quizzes will be in lecture on Thursdays
A few grading details

• 10 homework assignments, 3 points each
  • Can make up points from one homework with surveys

• 12 (graded) lab assignments, 2 points each
  • Two lowest lab scores will be dropped

• Written quizzes will be in lecture on Thursdays
  • We have sent out instructions for students who cannot attend Thursday lectures
A few grading details

- 10 homework assignments, 3 points each
  - Can make up points from one homework with surveys

- 12 (graded) lab assignments, 2 points each
  - Two lowest lab scores will be dropped

- Written quizzes will be in lecture on Thursdays
  - We have sent out instructions for students who cannot attend Thursday lectures
  - One written or coding quiz score will be dropped
A few grading details

• 10 homework assignments, 3 points each
  • Can make up points from one homework with surveys

• 12 (graded) lab assignments, 2 points each
  • Two lowest lab scores will be dropped

• Written quizzes will be *in lecture* on Thursdays
  • We have sent out instructions for students who cannot attend Thursday lectures
  • One written or coding quiz score will be dropped

• This class is *not* curved!
A few grading details

• 10 homework assignments, 3 points each
  • Can make up points from one homework with surveys

• 12 (graded) lab assignments, 2 points each
  • Two lowest lab scores will be dropped

• Written quizzes will be *in lecture* on Thursdays
  • We have sent out instructions for students who cannot attend Thursday lectures
  • One written or coding quiz score will be dropped

• This class is *not* curved!
  • *Collaboration*, not competition
The limits of collaboration
The limits of collaboration

- Everyone should give and receive help, because everyone benefits and learns
The limits of collaboration

- Everyone should give and receive help, because everyone benefits and learns

- There is only one rule:
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns

• There is only one rule:
  • \textit{Your code is yours, and yours only.}
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns

• There is only one rule:
  • *Your code is yours, and yours only.*

• This means that:
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns

• There is only one rule:
  • *Your code is yours, and yours only.*

• This means that:
  • You *cannot* copy or use code from anyone except your partner
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns

• There is only one rule:
  • Your code is yours, and yours only.

• This means that:
  • You cannot copy or use code from anyone except your partner
  • You cannot share your code with anyone except your partner
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns.

• There is only one rule:
  • *Your code is yours, and yours only.*

• This means that:
  • You *cannot* copy or use code from anyone except your partner
  • You *cannot* share your code with anyone except your partner

• Share and discuss *ideas*, not code
The limits of collaboration

• Everyone should give and receive help, because everyone benefits and learns

• There is only one rule:
  • *Your code is yours, and yours only.*

• This means that:
  • You *cannot* copy or use code from anyone except your partner
  • You *cannot* share your code with anyone except your partner

• Share and discuss *ideas*, not code
• Build good habits now!
Getting help
Getting help

• Discuss everything in the course, except exams, with your partner and your classmates
Getting help

- Discuss everything in the course, except exams, with your partner and your classmates
  - *Teaching* is the best way to learn
Getting help

• Discuss everything in the course, except exams, with your partner and your classmates
  • *Teaching* is the best way to learn

• Ask and answer questions on Piazza
Getting help

• Discuss everything in the course, except exams, with your partner and your classmates
  • *Teaching* is the best way to learn

• Ask and answer questions on Piazza

• Use the course staff! We’re here to help you learn
Getting help

• Discuss everything in the course, except exams, with your partner and your classmates
  • *Teaching* is the best way to learn

• Ask and answer questions on Piazza

• Use the course staff! We’re here to help you learn
  • Labs and office hours are the perfect time to talk to the lecturers, TAs, tutors, and lab assistants
Getting help

- Discuss everything in the course, except exams, with your partner and your classmates
  - *Teaching* is the best way to learn

- Ask and answer questions on Piazza

- Use the course staff! We’re here to help you learn
  - Labs and office hours are the perfect time to talk to the lecturers, TAs, tutors, and lab assistants
  - Lab assistants will also be available for *checkoffs* during labs
A few last thoughts
A few last thoughts

• Find all the course details and news on cs61a.org
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
    • 75% of students in this course receive As and Bs
A few last thoughts

- Find all the course details and news on cs61a.org

- The most important course policy is *not*:
  - Grading
    - 75% of students in this course receive As and Bs
    - There is no curve! All of you can get an A+
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is *not*:
  • Grading
    • 75% of students in this course receive As and Bs
    • There is no curve! All of you can get an A+
  • Cheating
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
    • 75% of students in this course receive As and Bs
    • There is no curve! All of you can get an A+
  • Cheating
    • There is a community of staff and students that want you to succeed, and will help you succeed
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
    • 75% of students in this course receive As and Bs
    • There is no curve! All of you can get an A+
  • Cheating
    • There is a community of staff and students that want you to succeed, and will help you succeed

• The most important course policy is learning
A few last thoughts

• Find all the course details and news on cs61a.org

• The most important course policy is not:
  • Grading
    • 75% of students in this course receive As and Bs
    • There is no curve! All of you can get an A+
  • Cheating
    • There is a community of staff and students that want you to succeed, and will help you succeed

• The most important course policy is learning
• Learn a lot, have fun, and welcome to 61A!
An Introduction to Programming

And, conveniently, an introduction to Python
Course organization
Course organization

- Every week will center around a theme, and have a specific set of goals.
Course organization

• Every week will center around a theme, and have a specific set of goals.

Introduction
Every week will center around a theme, and have a specific set of goals.

Introduction → Functions
Course organization

• Every week will center around a theme, and have a specific set of goals.
Course organization

- Every week will center around a theme, and have a specific set of goals.
Every week will center around a theme, and have a specific set of goals.

Course organization

Introduction → Functions → Data → Mutability

Objects
Course organization

- Every week will center around a theme, and have a specific set of goals.
Every week will center around a theme, and have a specific set of goals.
Course organization

• Every week will center around a theme, and have a specific set of goals.
Every week will center around a theme, and have a specific set of goals.

- This week (Introduction), the goals are:
Every week will center around a theme, and have a specific set of goals.

This week (Introduction), the goals are:

- To learn the fundamentals of programming

Course organization

- Introduction
- Functions
- Data
- Mutability
- Objects
- Interpretation
- Paradigms
- Applications

This week (Introduction), the goals are:
- To learn the fundamentals of programming
Course organization

- Every week will center around a theme, and have a specific set of goals.

- This week (Introduction), the goals are:
  - To learn the fundamentals of programming
  - To become comfortable with Python
What’s in a program?
What’s in a program?

• Programs work by manipulating values
What’s in a program?

- Programs work by manipulating values

- *Expressions* in programs evaluate to values
What’s in a program?

- Programs work by manipulating values

- Expressions in programs evaluate to values
  - Primitive expressions evaluate directly to values with minimal work needed
What’s in a program?

• Programs work by manipulating values

• Expressions in programs evaluate to values
  • Primitive expressions evaluate directly to values with minimal work needed

• Operators combine primitives expressions into more complex expressions
What’s in a program?

• Programs work by manipulating values

• Expressions in programs evaluate to values
  • Primitive expressions evaluate directly to values with minimal work needed

• Operators combine primitives expressions into more complex expressions

• The Python interpreter evaluates expressions and displays their values
What’s in a program?  

• Programs work by manipulating values

• *Expressions* in programs evaluate to values
  • *Primitive expressions* evaluate directly to values with minimal work needed

• *Operators* combine primitives expressions into more complex expressions

• The Python interpreter evaluates expressions and displays their values
Mathematical expressions
Mathematical expressions

\[ x + y \]
Mathematical expressions

\[ \frac{x}{y} \quad x + y \]
Mathematical expressions

\[ \sqrt{x} \]

\[ \frac{x}{y} \]

\[ x + y \]
Mathematical expressions

\( \sqrt{x} \)

\( \frac{x}{y} \)

\( \sin x \)

\( x + y \)
Mathematical expressions

\[ \text{sgn}(x) \quad \sin x \]

\[ \sqrt{x} \]

\[ \frac{x}{y} \quad x + y \]
Mathematical expressions

\[ sgn(x) \quad \sin x \]

\[ \sqrt{x} \]

\[ \frac{x}{y} \quad x + y \]

\[ x \mod y \]
Mathematical expressions

\[ sgn(x) \quad \sin x \]
\[ \sqrt{x} \]
\[ \frac{x}{y} \quad |x| \quad x + y \]
\[ x \mod y \]
Mathematical expressions

\[ sgn(x) \quad \sin x \]

\[ \sqrt{x} \quad \ln x \]

\[ \frac{x}{y} \quad |x| \quad x + y \]

\[ x \quad \text{mod} \ y \]
Mathematical expressions

\[
\lim_{x \to \infty} \frac{1}{x} \quad sgn(x) \quad \sin x \\
\sqrt{x} \quad |x| \quad \ln x \\
\frac{x}{y} \quad x \quad \text{mod} \ y \\
x + y
\]
Mathematical expressions

\[
\lim_{x \to \infty} \frac{1}{x}
\]

\[
\text{sgn}(x)
\]

\[
\sin x
\]

\[
\sqrt{x}
\]

\[
x^y
\]

\[
\ln x
\]

\[
x + y
\]

\[
\frac{x}{y}
\]

\[
|x|
\]

\[
x \mod y
\]
Mathematical expressions

\[ \lim_{x \to \infty} \frac{1}{x} \]

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

\[ \sqrt{x} \]

\[ |x| \]

\[ x \mod y \]

\[ \sin x \]

\[ x + y \]

\[ xy \]

\[ \ln x \]
Mathematical expressions

\[
\lim_{x \to \infty} \frac{1}{x}
\]

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

\[\binom{x}{y} \quad \binom{y}{x}\]

\[\sqrt{x}\]

\[\sin x\]

\[\ln x\]

\[\frac{x}{y}\]

\[|x|\]

\[x \mod y\]

\[x \cdot y\]
Mathematical expressions

\[
\lim_{x \to \infty} \frac{1}{x} \quad sgn(x) \quad \sin x
\]

\[
\sum_{i=1}^{n} i \quad \sqrt{x} \quad x^y \quad \ln x
\]

\[
\begin{pmatrix} x \\ y \end{pmatrix} \quad \frac{x}{y} \quad |x| \quad x + y
\]

\[
x \mod y
\]
Mathematical expressions

\[ \lim_{x \to \infty} \frac{1}{x} \]

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

\[ \binom{x}{y} \]

\[ \frac{x}{y} \]

\[ \sqrt{x} \]

\[ |x| \]

\[ x \mod y \]

\[ \sin x \]

\[ xy \]

\[ \ln x \]

\[ x + y \]
Call expressions
Call expressions

add ( 2, 3 )
Call expressions

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

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

operator \[ \underline{\ ]} \] \[ \underline{\ ]} \] operands
Call expressions

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

- In a call expression, the operator and operands themselves are expressions.
Call expressions

In a call expression, the operator and operands themselves are expressions.

To evaluate this call expression:

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

In a call expression, the operator and operands themselves are expressions.

To evaluate this call expression:

1. Evaluate the operator to get a function
Call expressions

In a call expression, the operator and operands themselves are expressions

To evaluate this call expression:

1. Evaluate the operator to get a function
2. Evaluate the operands to get its values
Call expressions

- In a call expression, the operator and operands themselves are expressions.

- To evaluate this call expression:
  1. Evaluate the operator to get a function.
  2. Evaluate the operands to get its values.
  3. Apply the function to the values of the operands to get the final value.

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

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

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

• What does this call expression evaluate to?
Nested call expressions

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

- What does this call expression evaluate to?
- What are the steps that the Python interpreter goes through to evaluate this expression?
The Power of Python

Shakespeare demo!