Welcome to Berkeley Computer Science!
The Course Staff

John DeNero
The Course Staff

John DeNero
The Course Staff

John DeNero
The Course Staff

John DeNero
The Course Staff

John DeNero

TAs run sections, labs, and also everything else
The Course Staff

John DeNero

TAs run sections, labs, and also everything else

Akihiro Matsukawa
Email: cs61a-tj

Hamilton Nguyen
Email: cs61a-tf

Phillip Carpenter
Email: cs61a-tl

Steven Tang
Email: cs61a-tx

Varun Pai
Email: cs61a-tj

Joy Cheng
Email: cs61a-te

Keegan Mann
Email: cs61a-tc

Allen Nguyen
Email: cs61a-tk

Stephen Martinis
Email: cs61a-ty

Andrew Nguyen
Email: cs61a-tg

Albert Wu
Email: cs61a-tl

Julia Oh
Email: cs61a-th

Shu Zhong
Email: cs61a-td
The Course Staff

TAs run sections, labs, and also everything else

Readers are your personal programming mentors
The Course Staff

TAs run sections, labs, and also everything else

Readers are your personal programming mentors

Lab Assistants ensure that you don’t get stuck
What is Computer Science?
What is Computer Science?

Systems
What is Computer Science?

Systems

Artificial Intelligence
What is Computer Science?

Systems

Artificial Intelligence

Graphics
What is Computer Science?

Systems

Artificial Intelligence

Graphics

Security
What is Computer Science?

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

...
What is Computer Science?

Systems
Artificial Intelligence
Graphics
Security
Networking
Programming Languages
Theory
Scientific Computing
...

Computer Vision
What is Computer Science?

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

...
What is Computer Science?

- Systems
- Artificial Intelligence
- Graphics
- Security
- Networking
- Programming Languages
- Theory
- Scientific Computing
- Computer Vision
- Planning
- Robotics
- Natural Language Processing

...
What is Computer Science?

- Systems
- Artificial Intelligence
- Graphics
- Security
- Networking
- Programming Languages
- Theory
- Scientific Computing

- Computer Vision
- Planning
- Robotics
- Natural Language Processing
- ...
What is Computer Science?

Systems
Artificial Intelligence
Graphics
Security
Networking
Programming Languages
Theory
Scientific Computing

Computer Vision
Planning
Robotics
Natural Language Processing

...
What is 61A?
What is 61A?

• A course about managing complexity
What is 61A?

- A course about managing complexity
  - Mastering abstraction
What is 61A?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Not about 1’s and 0’s
What is 61A?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Not about 1’s and 0’s

• An introduction to Python
What is 61A?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Not about 1’s and 0’s

• An introduction to Python
  ▪ All the features we really need: introduced today
What is 61A?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Not about 1’s and 0’s

• An introduction to Python
  ▪ All the features we really need: introduced today
  ▪ Understanding through implementation
What is 61A?

- A course about managing complexity
  - Mastering abstraction
  - Not about 1’s and 0’s
- An introduction to Python
  - All the features we really need: introduced today
  - Understanding through implementation
  - Programs that run other programs: meta-evaluation
What is 61A?

• A course about managing complexity
  ▪ Mastering abstraction
  ▪ Not about 1’s and 0’s

• An introduction to Python
  ▪ All the features we really need: introduced today
  ▪ Understanding through implementation
  ▪ Programs that run other programs: meta-evaluation

• A challenging course that will demand a lot of you
What is 61A?

Plone Conference. Photo courtesy of Kriszta Szita
Alternatives to 61A
Alternatives to 61A

CS 61AS: Self-paced 61A
Alternatives to 61A

CS 61AS: Self-paced 61A

CS 10: The Beauty and Joy of Computing
Course Policies
The purpose of this course is to help you learn
The purpose of this course is to help you learn

The staff is here to make you successful
Course Policies

The purpose of this course is to help you learn

The staff is here to make you successful

All the details are online:

http://inst.eecs.berkeley.edu/~cs61A/fa12/about.html
Collaboration
Collaboration

- Discuss everything with each other
Collaboration

• Discuss everything with each other
• **EPA**: Effort, participation, and altruism
Collaboration

- Discuss everything with each other
- **EPA**: Effort, participation, and altruism
- Homework can be completed with a partner
Collaboration

- Discuss everything with each other
- **EPA:** Effort, participation, and altruism
- Homework can be completed with a partner
- Projects *should* be completed with a partner
Collaboration

• Discuss everything with each other
• **EPA**: Effort, participation, and altruism
• Homework can be completed with a partner
• Projects *should* be completed with a partner
• Find a project partner in your section!
Collaboration

- Discuss everything with each other
- **EPA**: Effort, participation, and altruism
- Homework can be completed with a partner
- Projects *should* be completed with a partner
- Find a project partner in your section!

The limits of collaboration
Collaboration

• Discuss everything with each other
• EPA: Effort, participation, and altruism
• Homework can be completed with a partner
• Projects should be completed with a partner
• Find a project partner in your section!

The limits of collaboration

• One simple rule: don’t share code
Collaboration

• Discuss everything with each other
• **EPA**: Effort, participation, and altruism
• Homework can be completed with a partner
• Projects *should* be completed with a partner
• Find a project partner in your section!

The limits of collaboration

• One simple rule: don’t share code
• Copying project solutions is a serious offense!
Announcements
Announcements

• Next week, both section and lab will meet in the lab rooms.
Announcements

- Next week, both section and lab will meet in the lab rooms.
- Homework 1 is posted! All homework is graded on effort.
Announcements

• Next week, both section and lab will meet in the lab rooms.
• Homework 1 is posted! All homework is graded on effort.
• If you are on the waitlist, still complete assignments!
Announcements

• Next week, both section and lab will meet in the lab rooms.
• Homework 1 is posted! All homework is graded on effort.
• If you are on the waitlist, still complete assignments!
• Midterms are on 9/19 and 10/24. Final exam is on 12/13.
Announcements

• Next week, both section and lab will meet in the lab rooms.

• Homework 1 is posted! All homework is graded on effort.

• If you are on the waitlist, still complete assignments!

• Midterms are on 9/19 and 10/24. Final exam is on 12/13.

• Read the lecture notes before you come to lecture!
Announcements

• Next week, both section and lab will meet in the lab rooms.
• Homework 1 is posted! All homework is graded on effort.
• If you are on the waitlist, still complete assignments!
• Midterms are on 9/19 and 10/24. Final exam is on 12/13.
• Read the lecture notes before you come to lecture!
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

\[ 18 + 69 = 87 \]
Types of expressions

An expression
describes a computation
and evaluates to a value

\[ 18 + 69 = 87 \]
\[ \frac{6}{23} \]
\[ \sqrt{3493161} = 1873 \]
Types of expressions

An expression describes a computation and evaluates to a value.

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

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

An expression describes a computation and evaluates to a value.

\[ f(x) \]

\[ 18 + 69 \]

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

\[ \sin \pi \]

\[ \sqrt{3493161} \]

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

\[ | - 1869| \]

\[ \binom{69}{18} \]
Call Expressions in Python

All expressions can use function call notation

(Demo)
Anatomy of a Call Expression
Anatomy of a Call Expression

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

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

\textit{Operator}
Anatomy of a Call Expression

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

\underline{Operator} \quad \underline{Operand \ 0} \quad \underline{Operand \ 1}
Anatomy of a Call Expression

add ( 2 , 3 )

Operator
Operand 0
Operand 1

Operators and operands are expressions
Anatomy of a Call Expression

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

Operator \quad Operand 0 \quad Operand 1

Operators and operands are expressions

So they evaluate to values
Anatomy of a Call Expression

Operators and operands are expressions
So they evaluate to values

Evaluation procedure for call expressions:
Anatomy of a Call Expression

Evaluation procedure for call expressions:

1. Evaluate the operator and operand subexpressions

Operators and operands are expressions
So they evaluate to values
Anatomy of a Call Expression

Evaluation procedure for call expressions:

1. Evaluate the operator and 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))

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

mul
Evaluating Nested Expressions

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

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

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

24

26

add(2, mul(4, 6))

mul(4, 6)

mul(2, mul(4, 6))

mul

add

2

24
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

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

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

Output: 208
Data, Functions, and Interpreters
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

“The Art of Computer Programming”
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

“The Art of Computer Programming”

2

*Donald Knuth*
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

“The Art of Computer Programming”

2

Shakespeare’s 37 plays

*Donald Knuth*
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

“The Art of Computer Programming”

2

*Shakespeare’s 37 plays*

Donald Knuth

**Functions:** Rules for manipulating data
Data, Functions, and Interpreters

Data: The things that programs fiddle with

“The Art of Computer Programming”

2

Shakespeare’s 37 plays

Donald Knuth

Functions: Rules for manipulating data

Add up numbers
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

> “The Art of Computer Programming”
>
> 2
>
> *Shakespeare’s 37 plays*
>
> *Donald Knuth*

**Functions:** Rules for manipulating data

> *Count the words in a line of text*
>
> *Add up numbers*
Data, Functions, and Interpreters

Data: The things that programs fiddle with

“The Art of Computer Programming”

2

Shakespeare’s 37 plays

Donald Knuth

Functions: Rules for manipulating data

Count the words in a line of text

Add up numbers

Pronounce someone’s name
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

"The Art of Computer Programming"

2

*Shakespeare’s 37 plays*

*Donald Knuth*

*(Ka–N00TH)*

**Functions:** Rules for manipulating data

*Count the words in a line of text*

*Add up numbers*

*Pronounce someone’s name*
Data, Functions, and Interpreters

**Data:** The things that programs fiddle with

“The Art of Computer Programming”

2

Shakespeare’s 37 plays

Donald Knuth

(Ka–NOOTH)

**Functions:** Rules for manipulating data

Count the words in a line of text

Add up numbers

Pronounce someone’s name

**Interpreter:** An implementation of the procedure for evaluation