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
TAs hold discussion sections, labs, and office hours
The Course Staff

TAs hold discussion sections, labs, and office hours
The Course Staff

TAs hold discussion sections, labs, and office hours

Readers are your personal programming mentors
The Course Staff

John DeNero

TAs hold discussion sections, labs, and office hours

Readers are your personal programming mentors

Lab Assistants ensure that you don’t get stuck for too long
An Introduction to Computer Science
What is Computer Science?
What is Computer Science?

The study of
What is Computer Science?

What problems can be solved using computation,

The study of
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 design choices 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 design choices 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 design choices 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 design choices 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 design choices 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 design choices 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 design choices 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 design choices 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 design choices 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 design choices lead to effective solutions.

- Systems
- Artificial Intelligence
- Graphics
- Security
- Networking
- Programming Languages
- Theory
- Scientific Computing
- Games
- 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 design choices lead to effective solutions.

Systems
Artificial Intelligence
Graphics
Security
Networking
Programming Languages
Theory
Scientific Computing
Games
Robotics
Natural Language Processing
...
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 about 1’s and 0’s
What is This Course About?

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

• An introduction to Python
What is This Course About?

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

• An introduction to Python
  ▪ All the features we really need: introduced today
What is This Course About?

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

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

- A course about managing complexity
  - Mastering abstraction
  - Programming paradigms
  - Not about 1’s and 0’s

- An introduction to Python
  - All the features we really need: introduced today
  - Understanding through implementation
  - How computers interpret programming languages
What is This Course About?

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

• An introduction to Python
  ▪ All the features we really need: introduced today
  ▪ Understanding through implementation
  ▪ How computers interpret programming languages

• A challenging course that will demand a lot of you
What is This Course About?

Plone Conference. Photo courtesy of Kriszta Szita
Course Logistics and 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
Course Policies

The purpose of this course is to help you learn
Course Policies

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/fa13/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 (if you can)
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 (if you can)

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 (if you can)

The limits of collaboration

• One simple rule: Don’t share your code, except with partners
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 (if you can)

**The limits of collaboration**

- One simple rule: Don’t share your code, except with partners
- Copying project solutions is a serious offense!
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 (if you can)

**The limits of collaboration**

• One simple rule: Don’t share your code, except with partners
• Copying project solutions is a serious offense!
• We really do catch people who violate the rules
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 (if you can)

The limits of collaboration

• One simple rule: Don’t share your code, except with partners
• Copying project solutions is a serious offense!
• We really do catch people who violate the rules
  • We also know how to search the web for solutions
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 (if you can)

The limits of collaboration

• One simple rule: Don’t share your code, except with partners
• Copying project solutions is a serious offense!
• We really do catch people who violate the rules
  • We also know how to search the web for solutions
  • We let computers detect copying for us
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.

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

An expression describes a computation and evaluates to a value.

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

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

An expression describes a computation and evaluates to a value.

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

An expression describes a computation and evaluates to a value

$18 + 69$

$\frac{6}{23}$

$\sin \pi$

$\sqrt{3493161}$

$f(x)$

$| - 1869 |$

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

$\binom{69}{18}$
Call Expressions in Python

All expressions can use function call notation

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

\texttt{add ( 2 , 3 )}
Anatomy of a Call Expression

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

*Operator*
Anatomy of a Call Expression

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

- **Operator**: `add`
- **Operand 0**: `2`
- **Operand 1**: `3`
Anatomy of a Call Expression

\[ \text{add} \left( \begin{array}{c} 2 \\ \text{Operand 0} \end{array}, \begin{array}{c} 3 \\ \text{Operand 1} \end{array} \right) \]

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

Evaluation procedure for call expressions:

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
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

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

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

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
Evaluating Nested Expressions

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

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

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

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

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

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

```
208
```

```
mul
26
```

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

```
8
```

```
add(3, 5)
```

```
24
```

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

```
mul
```

```
4
```

```
6
```

```
add
2
```

```
add
3
```

```
add
5
```
Evaluating Nested Expressions

Operand 0 “subexpression”

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

Operand 0 “subexpression”

Expression tree
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”

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–NOOTH)*

**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