Welcome to Data C88C!

Instructor Team for Data C88C (and CS 61A)

Michael Ball (he/him) – **ball@berkeley.edu**

C88C instructor almost every semester since 2019

Research on language models and how people use them

Office hours 1pm-3pm Tuesdays in Warren 101B

John DeNero (he/him) – denero@berkeley.edu

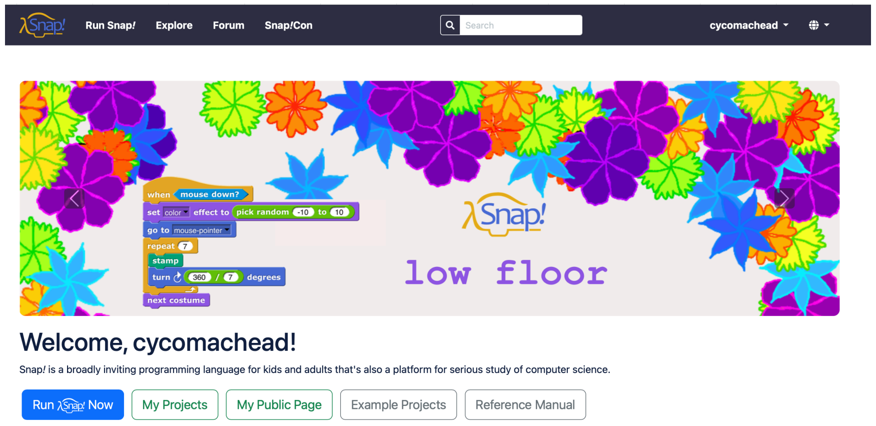
CS 61A instructor at least once per year since 2011

Research on language models and how people use them

Michael’s Research: Accessibility, Teaching CS, and Snap!

My recent research / work:

- How do we build accessible course software? How do we teach accessibility?
  - Accessibility: Building software such that it works for everyone, regardless of situation, disability, background, etc.
- Building blocks-based programming languages and communities.
  - I maintain the Snap! cloud infrastructure used for CS10 and students around the world.
- How to keep growing / training the awesome group of Berkeley TAs, tutors, and course staff?
- Formerly: Early engineer at Gradescope!



Welcome, cycomachead!

Snap! is a broadly inviting programming language for kids and adults that's also a platform for serious study of computer science.

Your Amazing Course Staff!

https://c88c.org/sp25/staff/

About the Course

Lecture, Videos, and the Textbook

Videos posted to c88c.org are essential viewing **before** coming to lecture. All of the course content will be covered in the videos.

The textbook, composingprograms.com, is written to be concise and useful. Its content is very similar to the videos.

Lecture Mon & Wed will review *the most important content* from the videos (but not all of it), work through examples, and discuss problem-solving strategies.

We will use the live Q&A Thread for each lecture: https://go.c88c.org/1

Problem-Solving Practice

Solving problems is an effective way to learn how to solve problems.

Lab: attendance is required-ish* (unless you're in the self-paced mega lab).

You'll get an update from your TA, discuss problems in groups, then work on the lab.

These prepare you for weekly **homework** assignments & 2 larger programming **projects**

Drop-in one-on-one assignment help (called "**office hours**" at Cal) starts next week.

This Week Only: If you didn't have lab (or didn't make it to lab), review the Discussion 0 handout and complete Lab 0 on your own.

My Office Hours (Time TBD): “Tea Hours” - Advising, high level questions, why study X? etc.

What does a "discussion" look like?

Expectation

https://engineering.berkeley.edu/students/academic-support/

Too Many Courses

https://www.microsoft.com/en-us/research/blog/grassroots-data-science-education-uc-berkeley/

Goal: Provide a great environment to learn how to solve problems through practice & *discussion*

I've seen small groups of Cal students do amazing things!

Lab (Starts This Week)

Unless you've elected the mega lab…

Labs are part “discussion” and part programming. Sometimes we’ll start with the discussion first, sometimes programming.

What happens during the discussion portion of lab?

- You're given a worksheet full of example problems to solve together & some instructions.
- The point is not just to solve those problems, but to learn how to solve similar problems.
  - Discussion problems aren't graded; you don't have to solve them all.
- Bring a laptop or tablet.

What happens after the discussion portion of lab?

- Work through the programming problems in the week's lab assignment. You're welcome to work with others and ask for help from your TA.

Attendance — It’s Important but not graded

We will take attendance but it’s not graded “for points”

You may attend sections other than those which are enrolled—however, we can’t exceed the room capacity.

For now, **don’t** switch *to the waitlist* on CalCentral. Open seats are fine to take.

Megasection – will be recorded and uploaded the next day.

Review https://c88c.org/

Asking Questions

?

Ed: You can reach all staff (private posts) and all students (public posts)

ball@berkeley.edu: Don't be surprised if I ask you to post on Ed

cs88@berkeley.edu: Goes to several staff members

What is This Course About?

A course about managing complexity

- Mastering abstraction
- Techniques for organizing complex programs

An introduction to programming

- Full understanding of Python fundamentals
- Large-er projects to demonstrate how to manage complexity

Different types of languages: Python & SQL

Should you take Data C88C?

According to the Syllabus: c88c.org/sp25/articles/about-c88c/

There is no formal programming-related prerequisite for Data C88C, but...

- Taking the course without any prior programming experience is typically quite challenging.
- Students who take the course without prior programming experience typically must spend more time to complete assignments.
- If you find it challenging to complete all of the required coursework in the first three weeks, we strongly recommend that you take another course first.
- We recommend you have taken DATA 8 or are concurrently taking DATA 8. This definitely helps with the programming practice.

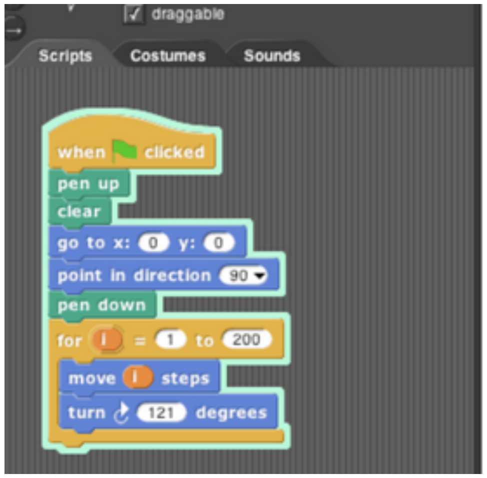
CS 10: The Beauty and Joy of Computing

Designed for students without prior experience

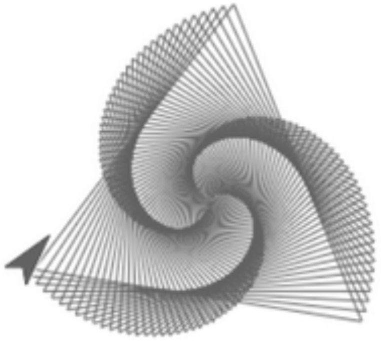
A programming environment created by Berkeley, now used in courses around the world and online

An introduction to fundamentals (& Python) that sets students up for success in Data C88C and CS 61A

More info: http://cs10.org/



```
when (flag) clicked
pen up
clear
go to x: 0 y: 0
point in direction 90
pen down
for i = 1 to 200
    move i steps
    turn 121 degrees
```



CS 61A

Data C88C is based on CS 61A, but covers only 3 out of 4 units worth of the content:

- Two programming projects (instead of four)
- Everything you need to know to continue on to CS 61B
- Omits the unit on how programs run other programs & a few advanced Python topics

Course Policies

Course Policies

Learning

Community

Details...

c88c.org/fa24/articles/about-c88c/

Collaboration

Working together is highly encouraged

- Discuss everything with each other; learn from your fellow students!
- Some projects can be completed with a partner

What constitutes academic misconduct?

- *Please* don't look at someone else's code!
  Exceptions: lab, your project partner, or **after you already solved the problem.**
- *Please* don't tell other people the answers! You can point them to what is wrong and describe how to fix it or show them a related example.
- *Please* don't ask ChatGPT or other similar tools to write code for you.
- Copying project solutions causes people to fail the course.

Build good habits now

Be careful with AI Tools

Things you can ask or do:

- “How do I quickly exit the Python interpreter?”
- “What is a REPL?”
- Autocomplete tools
- “Why does the following error message occur?”

Things You should be careful of:

- Show me how to do X in Python
  - This may be totally fine, but will often give you too much info.
- Show me an example of a factorial function that does ….
  (Because asking derivatives of assignment questions)

Things you cannot do!

- Directly ask AI to solve questions

Let's Stop Harassment & Discrimination

Disparaging remarks targeting a particular gender, race, or ethnicity are not acceptable.

From the Berkeley Principles of Community:

"We affirm the dignity of all individuals and strive to uphold a just community in which discrimination and hate are not tolerated."

From the EECS department mission:

"Diversity, equity, and inclusion are core values in the Department of Electrical Engineering and Computer Sciences. Our excellence can only be fully realized by faculty, students, and staff who share our commitment to these values."

EECS Student Climate & Incident Reporting Form: Informs the EECS department of any issues. You can also contact Antoine Davis (skauer@berkeley.edu) directly.

Expressions

Types of expressions

`An expression describes a computation and evaluates to a value`

$$\frac{6}{23}$$

$$\sin \pi$$

$$\log_2 1024$$

$$2^{100}$$

$$\boxed{f(x)}$$

$$\sqrt{3493161}$$

$$7 \bmod 2$$

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

$$\lim_{x \to \infty} \frac{1}{x}$$

$$|-1869|$$

$$\binom{69}{18}$$

Call Expressions in Python

All expressions can use function call notation

(Demo)

Terminals and Interpreters

Terminals: A “shell” — a program that lets you interact with your computer by typing commands

Getting good at this is useful **but not expected!**

Python Interpreter:

A program which *evaluates* Python code—as you write it—and immediately shows you the results.

SUPER useful! Use it liberally.

Tip: Up arrow key to recall previous statements.

python3 – this is often the exact name of the python interpreter program. It will also run other problems for us.

Demo