UC Berkeley EECS
Adj. Ass. Prof.
Dr. Gerald Friedland

Computational Structures in Data Science

Lecture #1: Welcome to CS88!

January 28, 2019 http://inst.eecs.berkeley.edu/~cs88

CS88 Team

Teaching Assistants


Alex Kassil
Email: alexkassil@berkeley.edu


Amir Shahatit
Email: ashahatit@berkeley.edu


Andrew Tan
Email: andrewtan@berkeley.edu


Brian Mi
Email: bmi@berkeley.edu


Jessica Gao
Email: gaojessicaping@berkeley.edu


John Yang
Email: john.yang20@berkeley.edu


Julia Yu
Email: juliayu@berkeley.edu


Lyric Yu
Email: lyricyu@berkeley.edu

Sophia Qin
Email: sophia.qin@berkeley.edu

Srinath Goli
Email: srig@berkeley.edu

CS88 Team

Tutors


Alec Kan
Email: alec.kan@berkeley.edu

Academic Interns


Akshatha Muralidhar


Andrew Cullen


Kevin Gu


Minos Park

CS88 Team - me

- Dr. Gerald Friedland (fractor@berkeley.edu)
  - 424 Saturdai Daj Hall (CITRIS)
  - http://www.gerald-friedland.org
  - Office hours: Mo 1:30-2:30 @ 424 SDH
  - Before/after class

Berkeley UNIVERSITY OF CALIFORNIA

LAWRENCE LIVERMORE NATIONAL LABORATORY

- Adjunct Assistant Professor, EECS UC Berkeley
- Principal Data Scientist, Lawrence Livermore National Laboratories

CS88 Team - me

Projects you might want to check out:

- http://mmcommons.org
  - Work with 100M images, 1M videos in your own Amazon instance.

- http://www.teachingprivacy.org
  - Creating teaching materials informing about data over sharing.

<Teaching Privacy>

Goals today

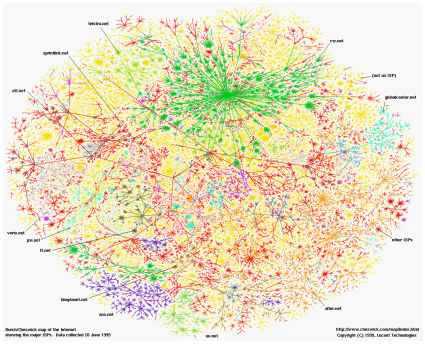
- Introduce you to
  - the field
  - the course
  - the team

- Answer your questions
- Big Ideas:
  - Abstraction
  - Data Type

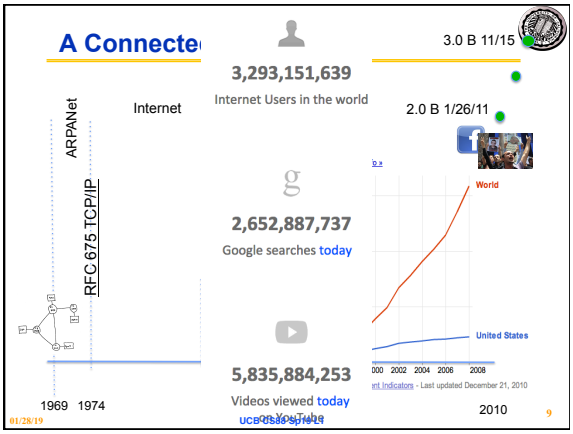
Data Science

Nearly every field of discovery is transitioning from "data poor" to "data rich"

Greatest Artifact of Human Civilization …

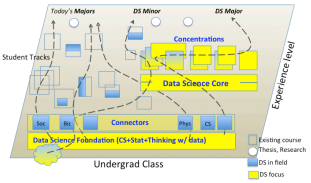


A Connecte



Data 8 – Foundations of Data Science

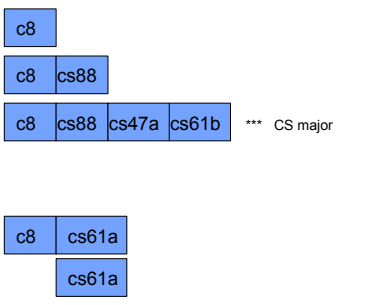
- **Computational Thinking + Inferential Thinking in the context of working with real world data**
- **Introduce you to several computational concepts in a simple data-centered setting**
  - **Authoring computational documents**
  - **Tables**
  - **Within Python3 and "SciPy"**



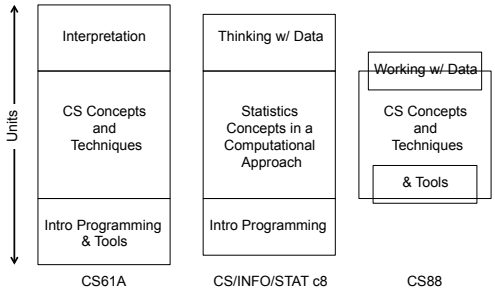
CS88 – Computational Structures in Data Science

- **Deeper understanding of the computing concepts introduced in c8**
  - **Hands-on experience => Foundational Concept**
  - **How would you create what you use in c8 ?**
- **Extend your understanding of the structure of computation**
  - **What is involved in interpreting the code you write ?**
  - **Deeper CS Concepts: Recursion, Objects, Classes, Higher-order Functions, Declarative programming, …**
  - **Managing complexity in creating larger software systems through composition**
- **Create complete (and fun) applications**
- **In a data-centric approach**

Pathways

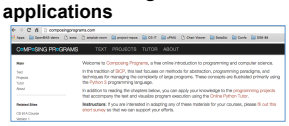


How does CS88 relate to CS61A ?



Course Structure

- 1 Lecture + 1 Lab/Discussion on Wednesday (!!!)
- Lecture introduces concepts (quickly!), answers why questions.
- Lab provides concrete detail hands-on
- Homework (10) cements your understanding
  - Out Monday, Due Sunday
- Projects (3) put your understanding to work in building complete applications
  - Maps
  - Hangman
  - Open Projects!

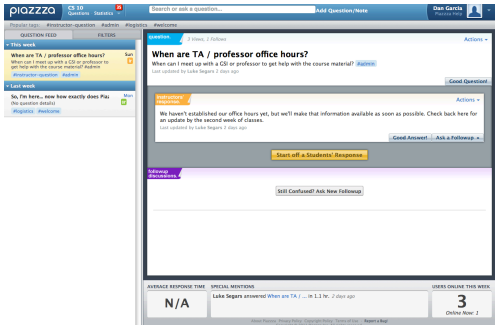


- Readings: http://composingprograms.com
  - Same as cs61a

Course Culture

- Learning
- Community
- Respect
- Collaboration
- Peer Instruction

Piazza for {ask,answer}ing questions



Where will we work?

- Datahub.berkeley.edu
- Your laptop
- Inst.eecs.Berkeley.edu

Pro-student Grading Policies

- EPA
  - Rewards good behavior
  - Effort
    - » E.g., Office hours, doing every single lab, hw, reading Piazza pages
  - Participation
    - » E.g., Raising hand in lec or discussion, asking questions on Piazza
  - Altruism
    - » E.g., helping other students in lab, answering questions on Piazza
- You have 2 "Slip Days"
  - You use them to extend due date, 1 slip day for 1 day extension
  - You can use them one at a time or all at once or in any combination
  - They follow you around when you pair up (you are counted individually)
    - » E.g., A has 2, B has 0. Project is late by 1 day. A uses 1, B is 1 day late

Abstraction

- **Detail removal**
  - **"The act of leaving out of consideration one or more properties of a complex object so as to attend to others."**
- **Generalization**
  - **"The process of formulating general concepts by abstracting common properties of instances"**
- **Technical terms: Compression, Quantization, Clustering, Unsupervized Learning**

Henri Matisse *"Naked Blue IV"*

Experiment

Where are you from?

Possible Answers:

- **China**
- **California**
- **The Bay Area**
- **San Mateo**
- **1947 Center Street, Berkeley, CA**
- **37.8693° N, 122.2696° W**

All correct but different levels of abstraction!

Abstraction gone wrong!

Detail Removal (in Data Science)

- **You'll want to look at only the interesting data, leave out the details, zoom in/ out…**
- **Abstraction is the idea that you focus on the essence, the cleanest way to map the messy real world to one you can build**
- **Experts are often brought in to know what to remove and what to keep!**

The London Underground 1928 Map & the 1933 map by Harry Beck.

The Power of Abstraction, Everywhere!

- **Examples:**
  - **Functions (e.g., sin x)**
  - **Hiring contractors**
  - **Application Programming Interfaces (APIs)**
  - **Technology (e.g., cars)**
- **Amazing things are built when these layer**
  - **And the abstraction layers are getting deeper by the day!**

We only need to worry about the interface, or specification, or contract NOT how (or by whom) it's built

Above the abstraction line

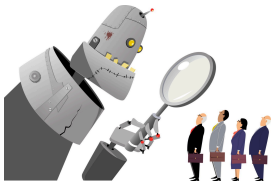
Abstraction Barrier (Interface) (the interface, or specification, or contract)

Below the abstraction line

This is where / how / when / by whom it is actually built, which is done according to the interface, specification, or contract.

Abstraction: Pitfalls

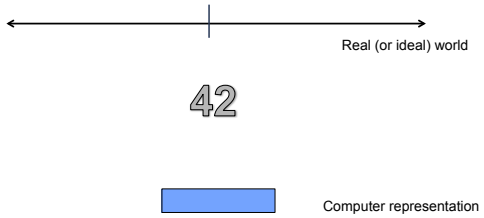
- Abstraction is not universal without loss of information (mathematically provable). This means, in the end, the complexity can only be "moved around"
- Abstraction makes us forget how things actually work and can therefore hide bias. Example: AI and hiring decisions.



- Abstraction makes things special and that creates dependencies. Dependencies grow longer and longer over time and can become unmanageable.

Abstraction in CS: Data Type

- What's this?



Data Types and Operations

- Set of elements
  - with some internal representation
  - E.g. Integers, Floats, Booleans, Strings, …
- Set of operations on elements of the type
  - e.g. +, *, -, /, %, //, **
  - ==, <, >, <=, >=
- Properties
  - Commutative, Associative, … , Closure (???)
- Expressions are valid well-defined sets of operations on elements that produce a value of a type

Lab and HW this week

- Lab will get you to where you have a *program development environment*
  - Even on your computer
- HW will give practice and explain subtleties of types, operators, and expressions
  - In a program development environment
- What's the difference between '==' and '=' ?

Thoughts for the Wandering Mind

A binary digit (bit) is a symbol from {0,1}.

- How many strings can you represent with N bits?
- Could you build a program that compresses all strings of N bits to strings of M bits (with M<N) such that you can go back to all original strings of length N? How or Why?