LEC 19

CSE 122

Victory Lap & Next Steps

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Slido Vote & chat with neighbors:

For science & to improve course timing, what was the most time-consuming/difficult assignment in CSE 122 for you?

Music: 122 25sp Lecture Tunes

Instructor: Brett Wortzman and Adrian Salguero

TAs:

Andrew	Diya	Logan	Steven
Anya	Elizabeth	Mahima	Yang
Brittan	Ivory	Medha	
Carson	Jack	Minh	
Christopher	Jacob	Nicole	
Colin	Ken	Samuel	
Dalton	Kyle	Shivani	
Daniel	Leo	Sreshta	

Announcements

- Lots of information posted this week!
  - Bob Bandes TA Award
  - Course Evaluations (due by Sunday, June 8th)
  - TA Feedback form (due Sunday, June 8th)
  - IPL closes today (Friday, June 6th)
  - Review Session Monday (June 9th 2:30pm – 5:00pm in JHN 120)
- Resubmission cycles R7 and R8 due Wednesday, June 11th
  - All assignments eligible for resubmission!
  - Also, info sent about R8 👀
- Quiz 2 grades out sometime this weekend
- Final exam: **Tuesday, June 10th 2:30 – 4:20 PM** in KNE 120 (here!)
  - Seating assignments are now posted

You Made It!

Lecture Outline

- Looking Back
- Looking Forward
- Thank You!

CSE 121 (or CS1) vs. CSE 122

CSE 121 / AP CS / IB CS / CS1 or **Other Programming Experience**

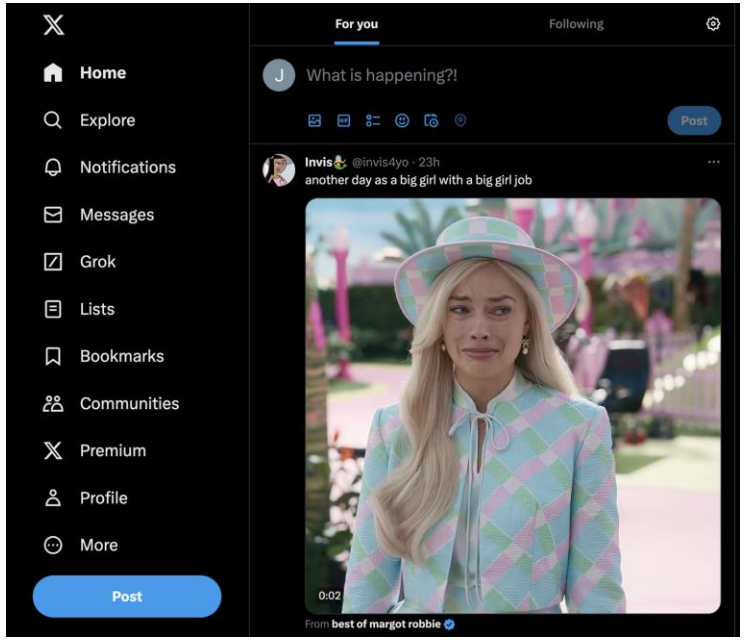
- Print statements
- Data types (`int`, `String`, `boolean`)
- Methods / Functions
  - Parameters
  - Returns
- Control structures
  - Loops
  - Conditionals
- Arrays & 2D Arrays
- **Computational Thinking** (language agnostic)

CSE 122 – Computer Programming II

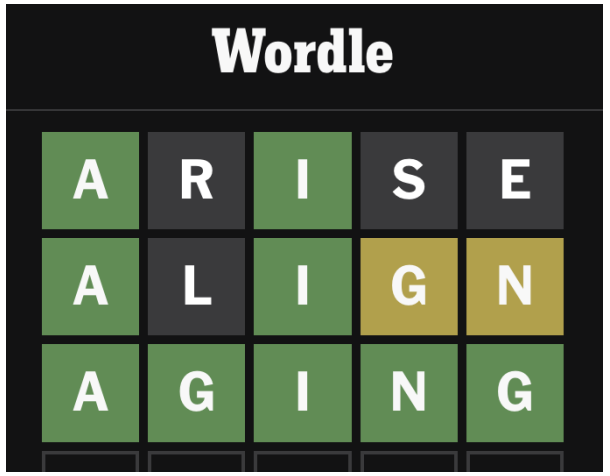
- Functional Decomposition
- File I/O
- Using data structures
  - List
  - Stacks / Queues
  - Sets
  - Maps
  - 2D Arrays
- Object Oriented Programming
  - Interfaces
  - Separation of Concerns

Why 122?

1. Build a strong foundation of data structures that will let you tackle the biggest problems in computing



122 Data Structures

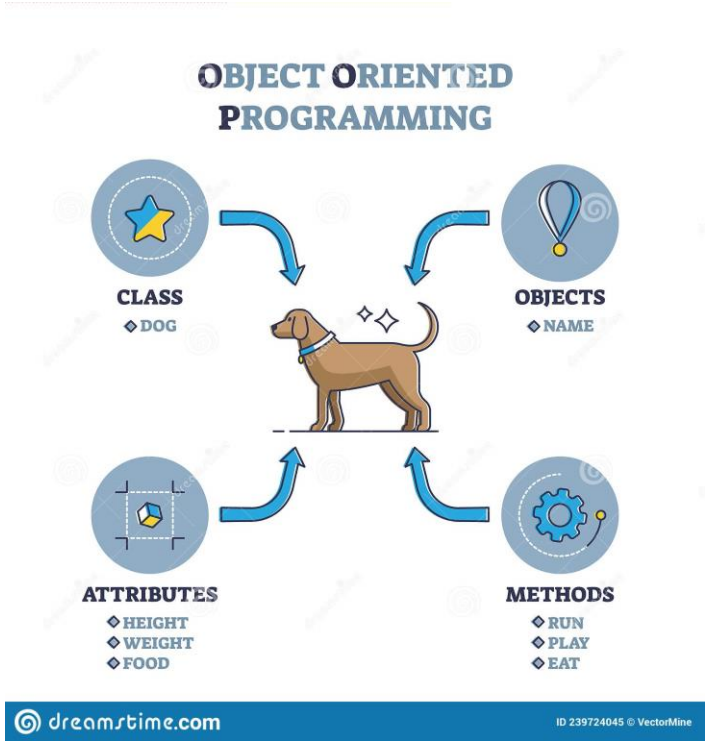


Why 122?

2. Learn an important structural pattern for representing **objects** in code to make our code more **reusable** and **maintainable** and **easier to understand.**

- Java is designed around this idea of objects. We haven't been leveraging that yet!

- Used in almost every real-world software project.



Review So Far

CS Concepts

- Problem Solving
- Functional Decomposition
- Debugging
- Testing
- Third Party Libraries*

Java Language

- File I/O
- Iterators and For-each Loops
- Exceptions
- Reference Semantics
- JUnit*

Data Structures

- ADTs
- Lists
- Stacks
- Queues
- Sets
- Maps

Java Collections

- Arrays / 2D Arrays
- ArrayList
- LinkedList
- Stack
- TreeSet / TreeMap
- HashSet / HashMap
- Interfaces for Collections

Object Oriented

- Instance variables
- Instance methods
- Interfaces
- Abstraction
- Encapsulation
- Client/Implementer

Lecture Outline

- Looking Back
- Looking Forward
- Thank You!

What Can Come Next?

- Some ideas
  - Work on a project
  - Learn a new (programming) language
  - Learn a new library
  - Take more courses
  - Explore CS beyond programming
- The general idea though is… whatever you want!
  - You've learned an extremely powerful set of skills, use it on what you are most interested in pursuing!

What Project?

- Add a Graphical User Interface (GUI) to an assignment
- Automate some boring tasks in your life
  - Maybe even automate writing code with good style?
- Organize and process data from your life (favorite quotes, your calendar, etc.)
- What are you currently doing that a computer could do?
- List of some project ideas (UW CSE alum)

What Language?

- Expanding your Java knowledge with a project is valuable. Or use a project to learn a new language!
- Pick a project, see what similar projects use! No wrong language to learn, certain tasks favor certain languages
  - iOS: Swift
  - Android: Java, Kotlin
  - Client-side web: Javascript (many frameworks to choose from)
  - Beautiful visuals: Processing
  - Data Processing + Machine Learning: Python
  - Data Management: SQL
  - Embedded systems: C / C++ / Rust

- Learn a new programming paradigm
  - Functional languages: Racket, Haskell, Scala, (now, Java 8!)

What Library?

Here are just a FEW examples. There is so much more!

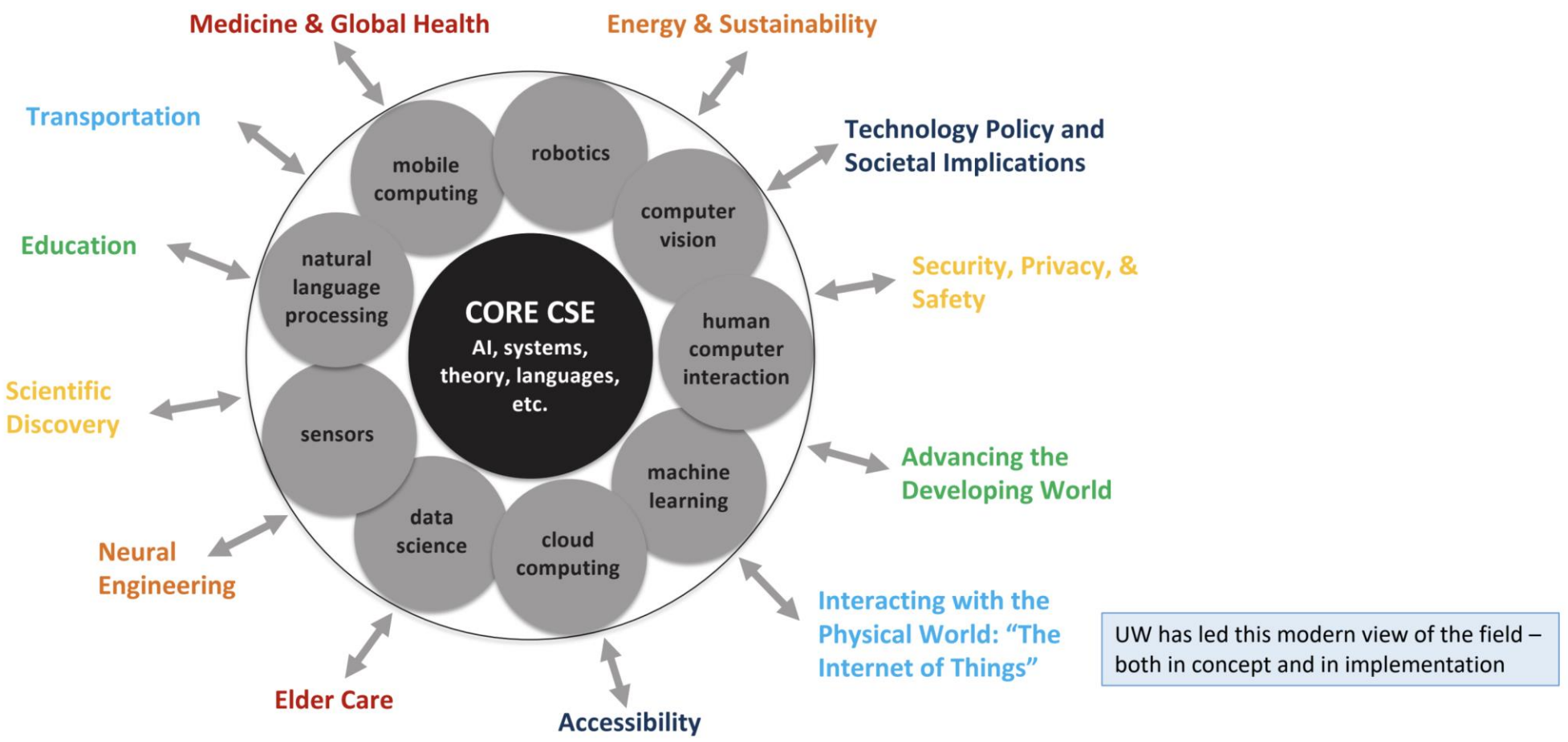
- Processing language
  - http://nlp.stanford.edu/software/
- Building games
  - http://lwjgl.org/
  - http://jbox2d.org/ (with physics!)
- Processing biological data
  - http://biojava.org/wiki/Main_Page
- Accessing Facebook data
  - http://restfb.com/
- Make a website backed by Java
  - https://www.jetbrains.com/help/idea/your-first-spring-application.html
- And more!

What Classes?

- CSE 123 is the most common next class. Continue the story, learn how data structures are implemented
- Other courses
  - CSE 154: Web Programming (HTML/CSS/Javascript)
  - CSE 163: Intermediate Data Programming (Python)
- **Large** set of CSE courses for *both* Allen School majors and students from all over UW campus. Many exciting courses, many (but not all) require CSE 123.
  - Allen School Majors
  - All UW Students
- Courses in Tech Related Majors: INFO, AMATH, HCDE, DXARTS, ...

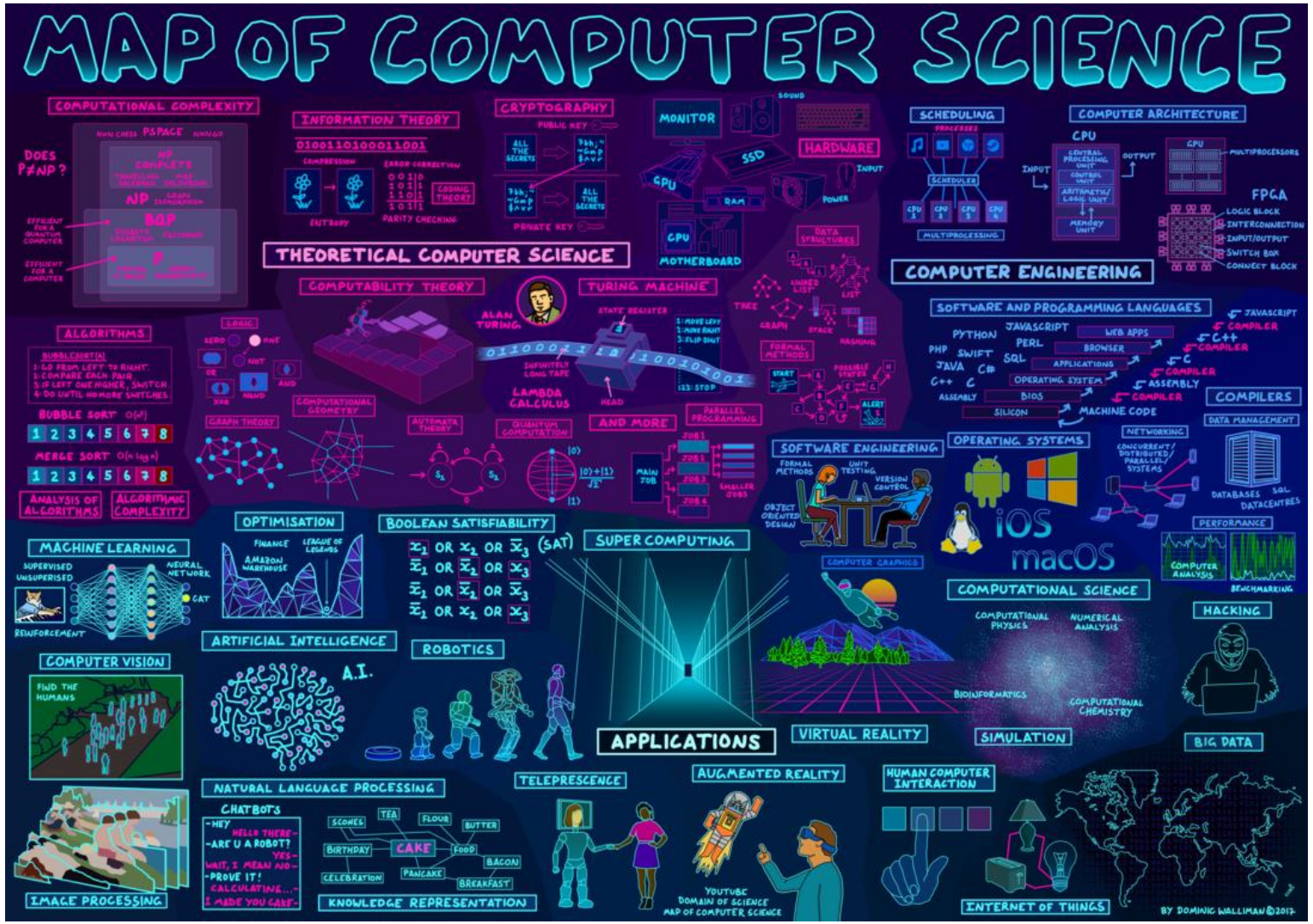
What is CSE?

The changing nature of the field: From smaller/faster/cheaper to tackling societal challenges



UW has led this modern view of the field – both in concept and in implementation

What is CSE?


MAP OF COMPUTER SCIENCE
THEORETICAL COMPUTER SCIENCE
COMPUTATIONAL COMPLEXITY
DOES P=NP?
PSPACE
NP
BQP
P
INFORMATION THEORY
CRYPTOGRAPHY
ALGORITHMS
BUBBLE SORT
MERGE SORT
ANALYSIS OF ALGORITHMS
ALGORITHMIC COMPLEXITY
COMPUTABILITY THEORY
TURING MACHINE
ALAN TURING
LAMBDA CALCULUS
AND MORE
COMPUTER ENGINEERING
HARDWARE
MONITOR
CPU
SSD
RAM
MOTHERBOARD
SCHEDULING
COMPUTER ARCHITECTURE
FPGA
SOFTWARE AND PROGRAMMING LANGUAGES
COMPILERS
OPERATING SYSTEMS
SOFTWARE ENGINEERING
MACHINE LEARNING
OPTIMISATION
BOOLEAN SATISFIABILITY
SUPER COMPUTING
ARTIFICIAL INTELLIGENCE
ROBOTICS
COMPUTER VISION
NATURAL LANGUAGE PROCESSING
KNOWLEDGE REPRESENTATION
IMAGE PROCESSING
APPLICATIONS
COMPUTER GRAPHICS
COMPUTATIONAL SCIENCE
VIRTUAL REALITY
SIMULATION
AUGMENTED REALITY
TELEPRESCENCE
HUMAN COMPUTER INTERACTION
INTERNET OF THINGS
BIG DATA
HACKING
BY DOMINIC WALLIMAN ©2017

Research Beyond Programming

Learn a new CS Topic

- Investigate how to best distribute relief funds
- Digitize basketball players
- Help deaf/hard-of-hearing people identify sounds
- Detect and prevent toxicity online
- Recognize disinformation online
- Make movies
- Improve digital collaboration
- Design algorithms that are more fair and better respect privacy
- Fix Olympic badminton
- And so much more!

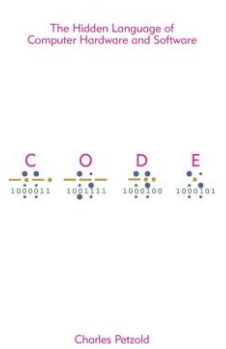
Attend Weekly Meetings

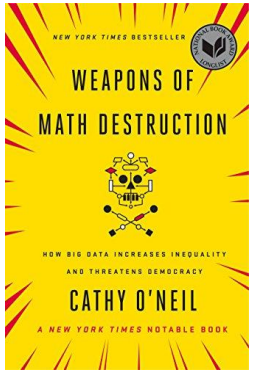
- Change – technologies for low-income regions
- Dub – human-computer interaction and design
- ComputingEd@UW – computer science education

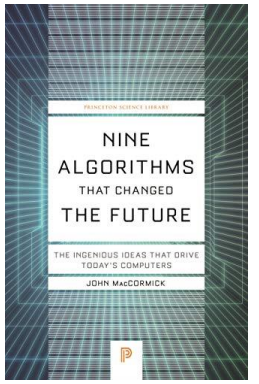
Registered Student Organizations (RSOs)

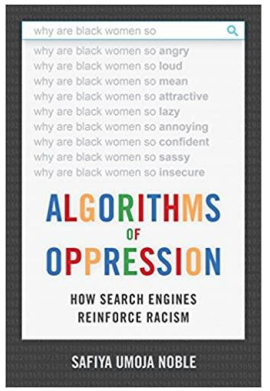
- Husky Coding Project – group projects, internship simulation
- DubHacks – student-run tech and entrepreneurship non-profit
- UW Game Dev Club – indie game development group
- Husky Robotics – robotics

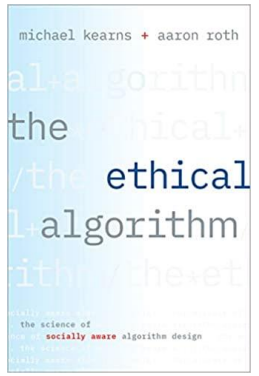
Read a Book! (links on pictures)

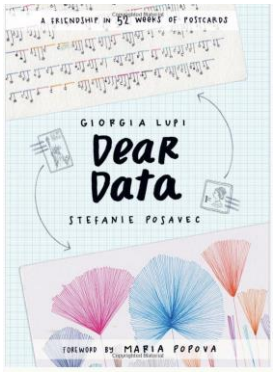












Lecture Outline

- Looking Back
- Looking Forward
- **Thank You!** 

Thank You! (Students)

- This is still a relatively new course! We are always looking for feedback on how to improve the class for you and for future students! Thank you for your patience and understanding as we develop everything. ☺
  - We <u>really</u> value your feedback!
  - Let us know what's working and what isn't working for you
  - Something that went well in another course? Tell us about it!
- **Please fill out the Course Evaluation by Sunday June 8th at 11:59 PM** to provide feedback about the course!

Thank You! (TAs)

Brett, Adrian, and you all couldn't have done this quarter without all of your amazing TAs! Thanks to them for running the course!

Ask Me Anything

AMA

sli.do #cse122