LEC 19

CSE 121

Victory Lap!

BEFORE WE START

Talk to your neighbors:

What is your go-to end-of-finals celebration?

Music: ❄ CSE 121 26wi Lecture Tunes ❄

Instructors: Miya Natsuhara

TAs:				
Amogh	Hayden	Anum	Sam	Shayna
William	Aki	Abdul	Ethan	Jesse
Johnathan	Spencer	Janvi	Jessica	Minh
Anant	Savannah	Navya	Paul	Cayden
Reese	Tamsyn	Ruslana	Carson	

Questions during Class?

Raise hand or send here

sli.do #cse121

Announcements, Reminders

- R7 (and R-Extra) due **Thursday, March 19th**
  - all assignments are eligible for resubmission!
- Today is the last day for IPL and instructor office hours
- Final Exam: **Wednesday, March 18th from 12:30 - 2:20pm**
  - review the Exam page of website (with policies & resources)
- TA-led review session: **Monday, March 16th from 4:30-7:00pm in CSE2 G20**
- Gumball & friends visit on **Monday, March 16th from 2:00pm-3:30pm** around Drumheller Fountain / Rainier Vista

Evaluations and Awards

Please give us feedback!

- **Course Evaluations** are due **Sunday, March 15th at 11:59 PM**
  - Lecture A Eval
  - Lecture B Eval
- **TA Evaluations** are *also* due **Sunday, March 15th at 11:59 PM**

Bob Bandes TA Award nominations open!

- thought your TA was goated? write them a nomination!
- fun fact: some of our faculty won the award when *they* were TAs!

You Made It!

Applications of Computer Science

or "What can I do with what I learned?" – outside of just "write code":

- Help deaf & hard-of-hearing people identify sounds
- Develop a programming language that celebrates the world's languages
- Build battery-free robots & put them on insects (and... track murder hornets?)
- Detect and prevent toxicity online & recognize disinformation
- Computational knitting & carpentry
- Create an interactive atlas of millions of refugee experiences
- Fix Olympic badminton & identify cheating in chess
- and so much more!

… including our assignments! (1/2)

- Computational Biology & Medicine (P2, P3)
  - in CSE: Chris Thachuk, Linda Shapiro, Sara Mostafavi, Su-In Lee, Luis Ceze
- Computational Art (C0, C1)
  - UW CSE has many unique intersections of CS + art!
  - "Cultural-Centric Computational Embroidery" (CSE + iSchool)
  - "Computational Illusion Knitting", "How to Knit Objects Weird"
  - "WasteBanned: Supporting zero waste fashion design"

… including our assignments! (2/2)

- Games & Graphics (C1, C3)
  - at UW: many labs in CSE and iSchool's GAMER group
  - fun fact: Foldit is a crowd-sourced game for protein folding
    - David Baker shared last year's Nobel Prize in Chemistry, in part for this!!
- Social Computing (P1)
  - at UW: Amy Zhang's Social Futures Lab + so much of iSchool
- Computer Security (C2)
  - in CSE: Franzi Roesner, David Kohlbrenner, Nirvan Tyagi
- and many side quests (in lecture, section, PCM): accessibility (e.g. UW CREATE), weather forecasting, chatbots, and lots of math

Future Courses

or "What can I do next?"

Non-majors

Course	Overview
CSE 154	Intro to web programming (several languages)
CSE 160	Intro programming, data analysis (Python)
CSE 163	Intermediate programming, data analysis (Python)
CSE 180	Introduction to data science (Python)
CSE 373	Data structures and algorithms (in Java)
CSE 374	Low-level programming and tools (C/C++)
CSE 412	Intro to Data Visualization
CSE 416	Intro. to Machine Learning
CSE 493E	Accessibility

More 12X!

Course	Overview
CSE 122	Data structures, object-oriented programming
CSE 123	More OOP, recursion

Majors

Course	Overview
CSE 311	Mathematical foundations
CSE 331	Software design/implementation
CSE 340	Interaction programming (mobile apps)
CSE 341	Programming languages
CSE 351	Hardware / Software Interface
CSE 480	Social impacts of computing

Related majors: Informatics, ACMS, HCDE, ECE, …

See: https://www.cs.washington.edu/academics/ugrad/current-students and https://www.cs.washington.edu/academics/ugrad/nonmajor-options/nonmajor-courses

Generalizing *beyond* Computer Science

Some of you said, ”I’m glad I took this class, but no more CS for me”

That’s <u>totally valid</u>!

Some lessons from this class that *could* apply more broadly:

- how to break big problems into smaller subproblems
- how to isolate what part of a system is broken
- attention to detail
- understanding basics of how software works
- how to learn (and reflect) effectively

Frequently Asked Questions

How can I get better at programming?

- Practice!

How can I learn to X?

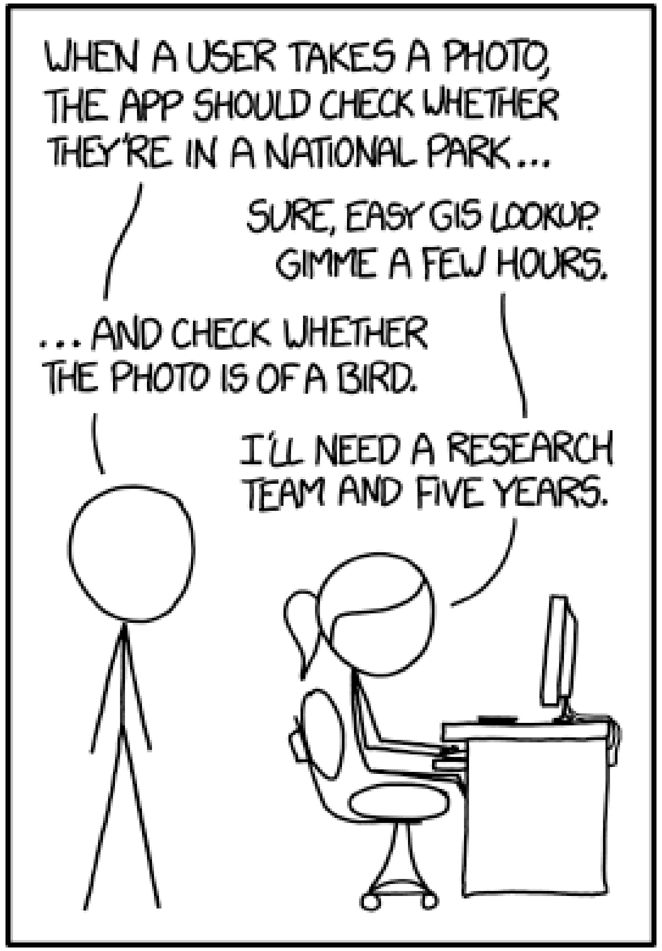
- Classes, books, videos, or self-learn!
- CS (as a field) has lots of free resources :)

What should I do next?

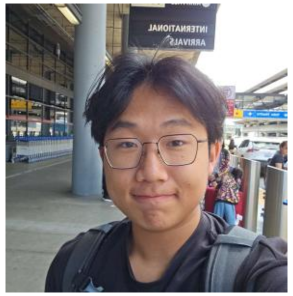
- Anything you're interested in!
- but: hard to tell what's easy and what's hard

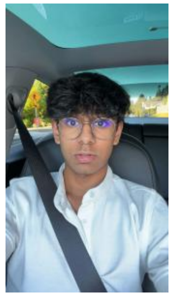
Should I learn another language? Which one?

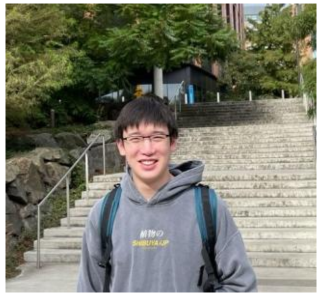
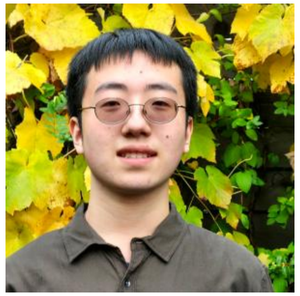
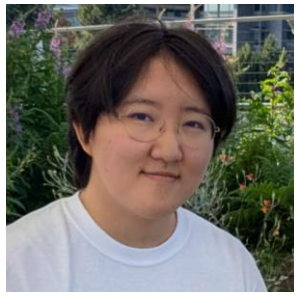
- That depends – what do you want to do?


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Thank your <u>fabulous</u> TAs!





Thank you!

Ask Me (Almost) Anything!

sli.do #cse121