CSCI 161
Introduction to Computer Science
Who Are We?

Let's get to one each other

- Preferred name?
- What is your class rank? (Freshman, sophomore, ...)
- What is your major, minor (if decided)?
- Prior programming experience, if any?
- What are you hoping to learn from this class?
Students taking this course range from those with no prior background in programming to those with substantial experience and prior coursework. Because this course is an introduction to computer science, I do not expect students to have had any prior experience in programming. The course will work best if we respect and welcome each other no matter what level of "readiness" we are at, and we all support one another in learning.

I will not tolerate behaviors that could negatively affect another student’s classroom experience. Such behaviors might include: making rude or condescending comments, snickering at others’ questions or comments, talking over other individuals, and so on. I reserve the right to withdraw a student from this class who is repeatedly exhibiting such behavior.
The goals of this course can only be accomplished in a setting of mutual respect, where ideas, questions, and misconceptions can be discussed with civility. As your instructor, I am committed to creating a classroom environment that welcomes all students, regardless of their identities (e.g., race, class, gender, sexual orientation, religious beliefs). I firmly believe that everyone in the class is fully capable of engaging and grasping the material, and that the world of computing is stronger when it includes the broadest possible set of perspectives. We all have unconscious biases, and I will try to continually examine my judgments, words, and actions to keep my biases in check and treat everyone fairly. I hope that you will do the same. If you feel comfortable, please let me know if there is anything I can do to make sure everyone is encouraged to succeed in this class.
About Me

- David Chiu (pronounced "chew")
  - Born: Taiwan; Raised in Ohio
  - Call me "Professor Chiu", "Dr. Chiu", or "David"

- Teaching: CS since 2002
  - ... as Masters student at Kent State (2002-04)
  - ... as PhD student at Ohio State (2005-10)
  - ... as Assistant Professor at WSU Vancouver (2010-14)
  - ... as Associate Professor at Puget Sound (2014-)
Return Policies for Computers?

- Ever pay attention to the return policies for computers?
  - Fine print from Best Buy:
    
    **30-Day Return Period**
    Thirty days from the date merchandise was received, refunds are available on the remainder of our products (see exceptions below).

    **14-Day Return Period**
    Fourteen days from the date merchandise was received, refunds are available on computers, monitors, notebook computers, projectors, camcorders, digital cameras, and radar detectors.

- From Costco:

  To keep our costs low, we must insist that the following be returned within **90 days** of purchase for a refund: televisions, projectors, computers, cameras, camcorders, touchscreen tablets, MP3 players and cell phones.
The Increasing Density of a Computer Chip

Moore's Law

- "Computer-chip density doubles (or, surface area halves) every 18 months"
  - Approximates that computing speed doubles every 18 months too!
  - Known as Moore's Law (end is nearing, though)
Why Study CS?

- **Reason 1:** Miniaturization of computers via Moore's Law
  - Real-world Impact: Today's problems are mostly data-driven
    - *New era of workers; tech and data literacy are needed*

- **Impact of Moore's Law**
  - Computers used to be rare, $$$
  - Now everywhere and cheap

- **Today and into the Future?**
  - Basic coding and understanding of tenets of computing becomes an essential skill like arithmetic
**Reason 2:** Highly interdisciplinary discipline

- **Communications**
  - Internet, chat, video streaming, social networks, GPS

- **Knowledge retrieval and Education**
  - Google, Wikipedia, Khan Academy, online college

- **Medicine and Public Health**
  - Healthcare software systems, drug design, flu prevention

- **Digital Media**
  - AR/VR, animation, music, and music synthesis, image processing

- **Business**
  - E-commerce, high-frequency trading
Why Study CS? (Cont.)

- **Reason 3:** Lucrative
  - PNW alone: Microsoft, Intel, HP, Amazon, Google, FB, Dell, ...

- **Mid-Career Salary**

  **Computer Scientist Salaries**
  924 Salaries  Updated Jan 21, 2020

  **Average Base Pay**
  $99,050 /yr

  **Additional Cash Compensation**
  Average $4,039
  Range $791 - $13,834

  How much does a Computer Scientist make?
  The national average salary for a Computer Scientist is $99,050 in United States. Filter by location to see... More

- IT Insider names UPS among top 20 colleges to go to get a job in tech
Outline

- Why Study Computer Science?
- Course Syllabus
- What Is Computer Science?
- Conclusion
Course Objectives

- Overarching goals:
  - Learn how to more abstractly and attack problems computationally
  - Reinforce critical thinking
  - Practice high levels of detail-orientation
  - Vessel: Do all that through the Java programming language

- Past students: "This field is all about solving puzzles."
Course Organization

- Textbook
  - Barnes and Kolling. "Objects First with Java." 5th edition or higher.

- Prerequisites
  - 3 years of high school math, or
  - MATH 110: Pre-calc or equivalent

- No previous programming experience is required or expected
Course Organization (Cont.)

- Course Web Page
  - [http://cs.pugetsound.edu/~dchiu/cs161](http://cs.pugetsound.edu/~dchiu/cs161)
  - Weekly schedule, labs, homework assignments, code examples, etc.

- Assignment and lab submission page:
  - [https://canvas.pugetsound.edu](https://canvas.pugetsound.edu)
How to Reach Me

- How to reach me (in order of preference):
  - Office: Thompson Hall 390B
  - Slack: univpugetsound.slack.com (signup with your pugetsound email)
  - Email: dchiu@pugetsound.edu

- Office Hours: Open-door
  - If door is cracked/open: Walk in Knock
  - If door is shut: Not around (or with someone else)
  - You can also make an appointment
Grading

- Breakdown
  - 8% Labs (11-12)
  - 35% Programming Assignments (7-8)
  - 15% Midterm Exam I
  - 17% Midterm Exam II
  - 20% Final Exam
  - 5% Participation (attendance, class participation)

Most important to your success:

Don't miss class
Budget for lots of time outside the class studying and programming
Ask lots of questions (first time most/all of you are seeing this stuff)
Come to tutoring and my office hours!
Labs (Bring Your Laptops!)

- Labs are meant to be a relaxed learning space
  - You are *paired* with a partner (different each week)
  - You are encouraged to talk amongst yourselves and ask for help

- Labs are *tutorial-like* at first...
  - Do this, now try this, try that
  - Write this code, test it out. Observe what happens
  - Covers heavily the topics we learned that week

- ... becomes incrementally more abstract
  - Write a program that does these 3 things
Lab Policies (Bring Your Laptops!)

- We meet weekly (Thursday) for lab
  - Paired-programming is *required*
  - Labs are always due *before* you leave
    - Finished way early? Must get graded before you leave*

- Labs are graded out of 0-2 scale
  - 2 = attended, (mostly) complete
  - 1 = attended, but program lacks depth
  - 0 = no submission, did not attend, or severely incomplete

- Get a head start! Download BlueJ tonight [www.bluej.org](http://www.bluej.org)
Homework Assignments

- Homework (Programming) Assignments
  - Less tutorial-like compared to labs
  - We try to solve real-world problems when possible

- Much more involved than labs
  - Culmination of concepts from weeks/months ago
  - A lot of "detective" work, leading to critical thinking
  - Work alone, but you can brainstorm with others!
    - Copy/pasting others' code = academic dishonesty!
Homework Assignments (Cont.)

- Over-budget for time spent outside of class
  - If you think it'll take you $Z$ hours, budget $1.5 \times Z$ hours
  - This is not a class in which you can start on homeworks the night before

- Be humble
  - People with prior coding background tend to do worse in this class
  - *(Like yours truly)*

- Larger programs: Don't start writing code *(Huh?)*
  - Plan things out by hand. It's easy to commit to the wrong path.
  - *(Hours/days of wasted time)*

- Don't be stubborn
  - Blow it up, and start with clean slate when appropriate
Exams (Hand-Written)

- Midterm exams take place on lab days

- Two midterms and a final exam:
  - All comprehensive, but weighted more heavily on newer materials
  - Allowed: Calculator and one full page of notes

- Rule of thumb:
  - I will not test on anything I did not specifically talk about in class
    - But things covered in class, but isn't found in the book is fair game
    - This *will* happen, so don’t miss class!
Classroom Participation

- Class should be **interactive**
  - Ask questions when you don't understand something
    - If questions are irrelevant to lecture, I'll be happy to answer it after class
  - I will pose lots questions each lecture
  - I will do lots of small-group work, when time permits

- Be careful how you present yourself
  - For many here, CS is completely new.
  - Try not to act condescending to others if you have a knack for this.
Things That Make Me Grumpy

- Class Disruption
  - Put phone on silent
  - No laptops, unless:
    - Lab day (Thursdays)
    - Permitted by SAA

- Cheating: Zero-Tolerance
  - I compare current and past assignments
  - Rules of thumb on collaboration:
    - Can do: Look on another student's screen to help them
    - Can't do: Look on another student's screen to help yourself

Image source: Grumpy Cat
Note-Taking for This Class

- Slides are always available on the course web page
  - After the lecture is over

- I do lots of code writing
  - When I write code on my laptop
    - No need to copy! I'll always provide the code we write in class
    - Just try to follow along, and understand the thinking behind the code
  - On the board (these are usually impromptu)
    - Write these down! You won't see these again

- I do lots of board-work: drawing/traces/illustrations
  - Copy these down
Backing Up Code

- Back up your code!

- Easiest way:
  - Use your UPS Google Drive account
    - Set it to sync automatically
  - When BlueJ asks where to place project folders
    - Always put them in the Google Drive folder
  - *I can help you set up this environment in lab*
CS Tutoring

- CS tutors are located in front of TH 409
  - Available everyday except Friday and Saturday
  - 4p - 6p

- Go here to see schedule:
  - No appointments, just sign-in
CS Tutoring Ground Rules

- Tutors are trained to **not** give away solutions
  - Don't badger them; it doesn't make them want to help you more
  - Tutors will **not** write code for you

- Don't expect to do your entire assignment in one night with a tutor
  - Tutors have not done the assignments we assign **you**
  - Not their problem that you didn't start earlier

- Tutors are students like you
  - Volunteering *their* time, so be nice

- If you have not been treated appropriately by a tutor
  - You're asked to contact any of the CS profs immediately (... and vice versa!)
If you have a physical, psychological, medical, or learning disability, contact:

- Peggy Perno
- Office of Student Accessibility and Accommodation (SAA)
- 105 Howarth Hall
- 253.879.3395

Communications with SAA is confidential
Computer Rental

- Laptop rental for a semester?
- Send me an email (dchiu@pugetsound.edu):
  - Windows or Mac?
  - Description of need

- These are not considered to be "needs":
  - My other computer is for gaming
  - My other computer is too slow and I want a new one
  - I want to separate work for this class from my other class
Outline

- Why Study Computer Science?
- Course Syllabus
- What Is Computer Science?
- Conclusion
What Is Computer Science (CS)?

- Computer science is the *study of computers, right?*

"Computer Science is no more about computers than astronomy is about telescopes."
- *CS pioneer, Edsger Dijkstra (1930-2002)*

- What computer science is *not*
  - Knowing how to use computers competently
  - Building or repairing computers
Lots of definitions of CS out there, but the central theme is this:

- Centers around *problem solving* through *algorithms*.
- This class is more about solving puzzles than anything else

Better get to know what *algorithms* are first...

- So, what are algorithms?
- Thoughts?
- Guesses?
Okay, What Are Algorithms?

An *algorithm* is a well-defined **sequence of instructions** such that:

- Given input (or not),
- The execution of each instruction (*without question*)
- Produces a *desired output*, or an *error*.

**Examples of algorithms in real life:**

- Steps on how to build a bird house
- Instructions on how to install a water filter
- Recipe to bake an apple pie
- Instructions on how to draw pacman... (yesterday's lab)
Yesterday's Lab

Algorithm

1. Create a new Circle named body
2. Create a new Circle named eye
3. Create a new Circle named powerball
4. Create a new Triangle named mouth
5. Make body visible
6. Make eye visible
7. Make mouth visible
8. Make powerball visible
9. Change body color to "yellow"
10. Change body size to 25
11. Change body size to 75
12. Move body down
13. Move body down
14. Move body up
15. Move body up
16. Move body up
17. Move body up
18. Change eye color to "black"
19. Change eye size to 5
20. Slow-Move eye horizontally by 58
21. Change mouth size to 75 (height) and 75 (width)
22. Move mouth horizontally by 10
23. Move mouth vertically by 40
24. Change mouth color to "white"
25. Change powerball color to "magenta"
26. Change powerball size to 20
27. Slow-Move powerball horizontally by 30
28. Move powerball down
29. Move powerball down

Input: None

Output:

Computer: the thing that executes algorithms.
Characteristics of algorithms

- Algorithms solve a problem
  - But some problems can't be solved algorithmically

- Algorithms are repeatable
  - Always produces the same result
  - (It's what makes us a science)

- There could be multiple algorithms that solve the same problem
  - Some are faster than others
  - Some are produce better results

- Algorithms may not always stop
  - Sometimes it's by mistake, sometimes not
Definition: What Is Computer Science?

- Computer Science is the *study of algorithms*, including their
  - Formal and mathematical properties,
  - Hardware realizations,
  - Linguistic realizations, and
  - Applications

*Definition from: Schneider and Gersting. An Invitation to Computer Science.*

- Computers are just tools (hardware) that carries out algorithms!
  - "Computer Science" is a misnomer
    - It's like calling Chemistry, "Beaker Science"
Two fundamental issues in computer science theory:

- Given a problem, determine its:
  - Decidability: Can the problem be solved algorithmically?

What are some **decidable** problems?

What are some **undecidable** problems?
Two fundamental issues in computer science theory:

- Given a problem, determine its:
  - Decidability: Can the problem be solved algorithmically?
    - What's the shortest path from city A to city B?
      » Decidable!
    - Will a program running on your computer ever stop?
      » Undecidable! (Can't be solved algorithmically)
Two fundamental issues in computer science theory:

- Given a problem, determine its:
  - **Time complexity ("hardness")**: If problem is decidable, can it be solved in a reasonable amount of time (*i.e.*, before end of humanity)?
    - Finding the shortest route between City A and City B is easy.
    - Making the perfect move in chess is **hard**.
    - If you were given a set of pieces in Tetris, can you survive? **Hard**.
Computer science is the *study of algorithms*, including their

- Formal and mathematical properties,
- Hardware realizations,
- Linguistic realizations, and
- Applications

*Definition from: Schneider and Gersting, An Invitation to Computer Science.*
Now we're concerned with, "How do we execute an algorithm?"

- Computers used to be a human occupation!

Earliest documented case of Human Computers (1759):

- Predicted the date of Halley's Comet's return
  - Read more here http://adsabs.harvard.edu/full/1993JHA....24....1W
The First Mechanical Computer

- Problems of human computers: inefficient, unreliable, & error-prone
  - Want something automatic, tireless, and durable - an engine!
- Charles Babbage (1791-1871) and Ada Lovelace (1815-1852)
  - Invented the Difference Engine, a mechanical calculator
  - Drew up plans for the Analytical Engine, but was never built
  - Watch: Calculating Ada documentary

Part of the Difference Engine, Science Museum in London

Ada Lovelace: Published the first program (for the analytical engine)
The First **Electronic Computer**

- ENIAC: Electronic Numeral Integrator and Computer (1946)
  - Constructed and operated at UPenn

- Processing rate of 5 kilo-hertz (KHz)
  - (1 KHz = 1000 cycles per second)
  - That is, at best, 5000 instructions is executed per second

- My Samsung Galaxy phone runs at 1.5 GHz
  - 300,000 times faster
  - Oh and my phone has 8 brains (or cores)
Computer science is the study of algorithms, including their

- Formal and mathematical properties,
- Hardware realizations,
- Linguistic realizations, and
- Applications

"The computer is an extremely fast moron. It will, at the speed of light, do exactly what it is told to do--no more, no less."

- Dr. Grace Murray Hopper
  Computer Science pioneer
The computer is an "extremely fast moron"

"..extremely fast.."

- Today's computers can do \(=\)trillions of simple operations per second!
  - Fastest computer to date is the *Tian-He 2* in China
  - Can perform 1.0 quadrillion "operations" per second
The computer is an "extremely fast moron"

"... moron"

- Computers lack insight
- You must tell it what to do in great detail, and its operations are very primitive

This is what makes programming both frustrating and super rewarding!
Computers are also reliable.

**Reliable:**

- They never tire, disobey instructions, or complain!
- Most important
  - Computers *never* make mistakes!
  - Instructions are never done out of order.
  - Implies that mistakes (called *bugs*) are only ever made by the programmer.
Just how *primitive* must these instructions be to a computer?

Consider this: Compute $D = A + (B - 4 \times C)$

Instructions to do $D = A + (B - 4 \times C)$:
- **LOAD** Contents of A
- **LOAD** Contents of B
- **LOAD** Contents of C
- **MULT** C and 4 and **STORE** in TMP
- **SUB** TMP from B and **STORE** in TMP2
- **ADD** A to TMP2 and **STORE** in D

Even this is not primitive enough!

I don't understand English.
I speak 1s and 0s.
Just how *primitive* must these algorithmic operations be?

Consider this: Compute $D = A + (B - 4 \times C)$

**Instructions to do $D = A + (B - 4 \times C)$:**

01010101110111010010101000101
0101011101101001010100010101
0101010111011001010100010101
01110110110010101000101010101
11011101001010100010101010101
01110110101010001010101010111
01110110101010001010101011101

**What each sequence means**

- **LOAD** Contents of $A$
- **LOAD** Contents of $B$
- **LOAD** Contents of $C$
- **MULTIPLY** $C$ and $4$ and store in TMP
- **SUB** TMP from B and store in TMP2
- **ADD** A to TMP2 and store in D

"Ah, now we're talking!"
(I only consume binary instructions)
How do you tell (i.e., program) a computer to do what you want?

Predict the results of the 2020 presidential election, kthx

Humans
Speak natural language
Abstract operations

Computers
Speak Binary
Primitive operations

Programming Languages
(The happy medium)

Java
C
C++
Swift
Python
PHP
Perl
Ruby
JS
Ada
Go
...

Don't understand! I speak in 1s and 0s! Oh, and can you give me primitive step-by-step instructions on what to do?
Algs: Linguistic Realizations (Cont.)

Hmm… how do I solve this new problem?

Algorithm Design

Tell computer about your algorithm (i.e., you program)

Algorithm (or code, or program)

Translate code to 0s and 1s

(known as "compiling")

Computer executes your algorithm. Solves problem, or throws error

Computer Scientist

Every software program is just lines and lines of instructions bossing a computer around!

Written by humans using Java, C, C++, Python, etc.
Compiled down to instructions in 1s and 0s
Fed to the computer for processing
Computer science is the *study of algorithms*, including their
- Formal and mathematical properties,
- Hardware realizations,
- Linguistic realizations, and
- Applications

*Definition from: Schneider and Gersting. An Invitation to Computer Science.*
Next "sea-change" areas in our lifetime?

- Transportation: autonomous cars (death of traffic lights)
  - MIT research: https://www.youtube.com/watch?v=kh7X-UKm9kw
- Virtual reality (VR) and augmented reality (AR)
  - Magic Leap glasses
- Smart "things": homes, AI assistants, etc.
Full circle now..

**Computer applications (programs)** start with an idea...

- Is the idea solvable with algorithms?
  - If not, stop now!

- Can the algorithm be processed quickly?
  - If so, program the algorithm, and improve on it over time!

- Considerations:
  - What language should I use?
  - What hardware should I run the program on?
  - Why is this program useful?
Outline

- Why Study Computer Science?
- Course Syllabus
- What Is Computer Science?
- Conclusion
Conclusion

- You should be able to...
  - Summarize why CS is important to know, even at a basic level
  - Define what CS and algorithms are
  - Understand the role of the computer scientist: we're problem solvers!

- Insights gleaned:
  - Is a problem decidable? Is it solvable in a reasonable amount of time?
  - Why have programming languages? What does a compiler do?
  - Computer is a tool, not a field of study.

- Next time: What Is Object-Oriented Programming?
Lab 1 post-mortem

- A Bird's Eye Tour of BlueJ
- Classes vs. Objects in Java
  - Difference between class menu items and object menu items?
  - What does it mean when a class is "greyed out" in BlueJ?
- The coordinate system as we know it seems... different?
- Self-Check - You should know:
  - Save/open and zip/submit BlueJ project files
  - Create and manipulate objects using the tool
  - Every (shape) object is identified by a name that the programmer assigns
  - Does instruction order matter?
Last week:

- Defined "Algorithms"
- In the process of defining "Computer Science"

Four properties of algorithms:

- Formal
  - Decidability and complexity of a problem?
- Hardware realizations
  - Foundations of hardware? Babbage/Ada's Analytical Engine

Today:

- Finish other two properties of algorithms
- Peeking at code structure: Circle class