CS 134:
Wrap Up

Announcements & Logistics

- **Lab 9 Grading:** Coming soon! Hope to return early next week
- **Final exam:**
  - **Wed May 18 @ 1pm in TCL 202**
  - **Sun May 22 @ 9:30am in Bronf Aud (aka Wach B11)**
    - **Reduced distractions/extra time Wach 015 (come to B11 first)**
  - **2 hour closed book exam**
  - Cumulative w/ more weight on topics post-midterm topics
  - Practice problems are posted; review lecture slides, jupyter notebooks, HWs, and labs
  - Format will be very similar to midterm
- **Review session: Tue, May 17, 8-9:30pm, Room TPL 203**
  - Very informal, come ask us questions
- **Office hours next week: TBD (check webpage!)**

Last Time

- Reviewed OOP concepts using Python and Java as examples
  - A **class** vs an **instance** of the class
  - **Attributes** (instance variables) and __slots__
  - **Accessor** and **mutator** methods: getters, setters
  - **Scope**: public, private and protected (or _ and __ in Python)
  - Special methods and operator/function overloading

Today

- Summarize **main topics** covered in CS 134 this semester
- Complete **course evals**
    - We'll end lecture early to leave time for you to fill out evals

CS134 in a Nutshell

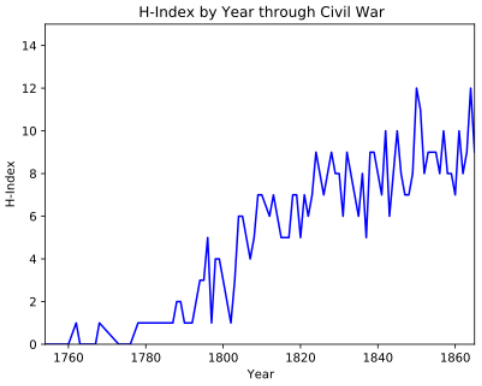
- We have covered many topics this semester!
- We started out learning the basics of Python and programming in general
- **Pre-midterm**
  - **Types & Operators** (int, float, %, //, /, concatenation, etc)
  - **Functions** (variable scope, return vs print, defining vs calling functions)
  - **Booleans and conditionals** (if elif else)
  - **Iteration**: for loops, while loops, nested loops, accumulation variables in loops
  - **Sequences**: strings (string methods, in/not in, iteration, etc) , lists (list methods, append, extend, etc), ranges, tuples, lists of lists
  - **File reading**: with … as , strip(), split()
  - **Mutability** and **aliasing**

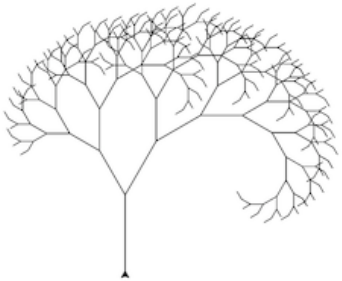
CS134 in a Nutshell

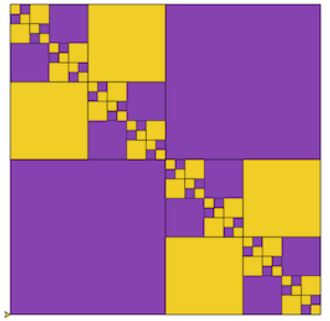
- Then we moved on to more advanced CS topics
- **Post-midterm**
  - **Data structures**: More tuples, dictionaries, sets
    - **Sorting** data with key functions
  - **Recursion**: recursive methods and classes
    - **Graphical recursion** with **turtle** graphics
  - **Classes, Objects, and OOP**
    - attributes, __slots__, special methods, getters, setters, inheritance
    - "Bigger" OOP Examples: Tic-Tac-Toe, Boggle, Linked list
  - **Advanced topics**:
    - Efficiency, Searching and sorting, Iterators, Python vs. Java

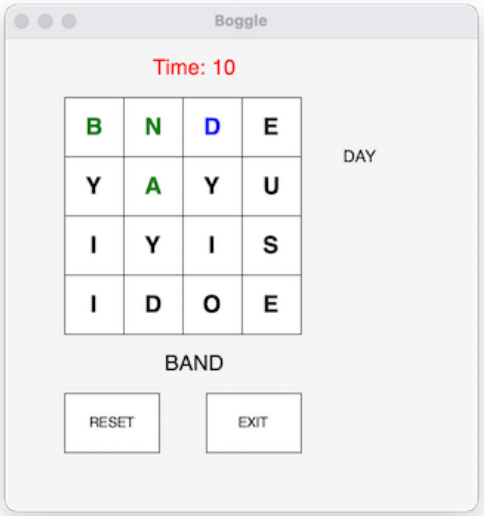
Labs

- Hello, World!
- Day of the week (conditionals)
- Word puzzles (strings and loops)
- Voting algorithms (lists and loops)
- Debugging
- Supreme Court (dictionaries and plotting)
- Recursion
- Autocomplete (classes and methods)
- Boggle (OOP, more classes and inheritance)
- Selection sort (Java)



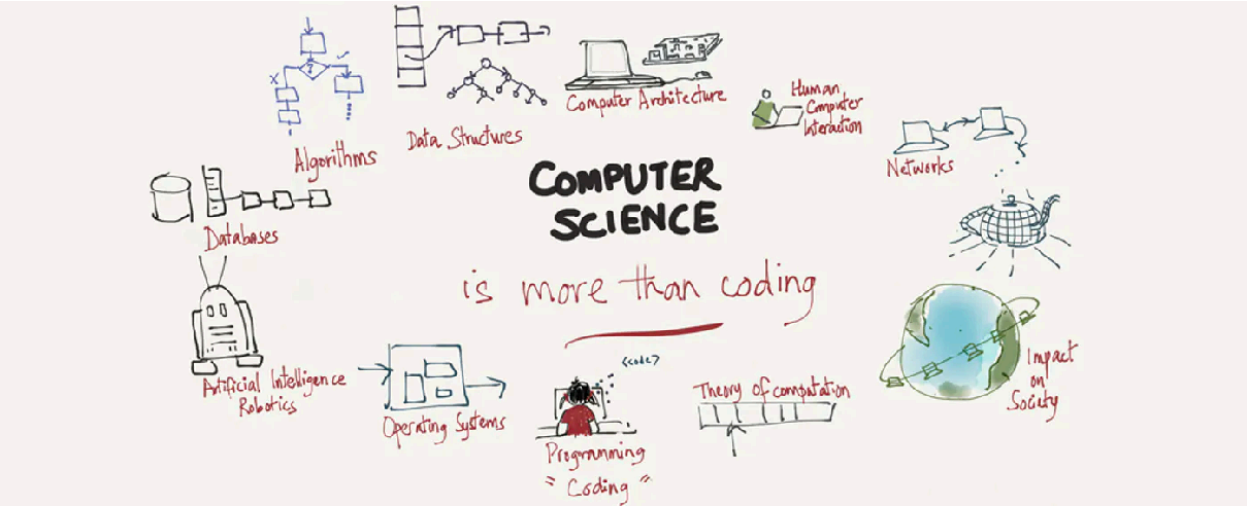






Takeaway: What is Computer Science?

- Computer science ≠ computer programming!
- Computer science is the study of what computers [can] do; programming is the practice of making computers do useful things
- Programming is a big part of computer science, but there is much more to CS than just writing programs!
- Another part of CS is computational thinking



https://www.edsurge.com/news/2015-12-02-computer-science-goes-beyond-coding

Take away: Computational Thinking

- Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. We can then present these solutions in a way that a computer, a human, or both, can understand.
- Four pillars of CT:
  - **Decomposition** - break down a complex problem or system into smaller, more manageable parts
  - **Pattern recognition** – look for similarities among and within problems
  - **Abstraction** – focus on important information only, ignore irrelevant details
  - **Algorithms** - develop a step-by-step solution to the problem, or the rules to follow to solve the problem
- A computer can performs billion of operations per second, but computers only do exactly what you tell them to do!
- In this course we ~~will learn~~ **learned** how to 1) use CT to develop algorithms for solving problems, and 2) implement our algorithms through computer programs

Goals from Lecture 1

- Abstraction and modularity
- Representing knowledge with data structures
- Iteration and recursion as computational tools
- Divide and conquer problem solving strategies
- Testing and debugging
- Organizing and dealing with data
- Plotting and visualizing data
- Playing with python graphics
- Transitioning from Python to Java (and beyond!)

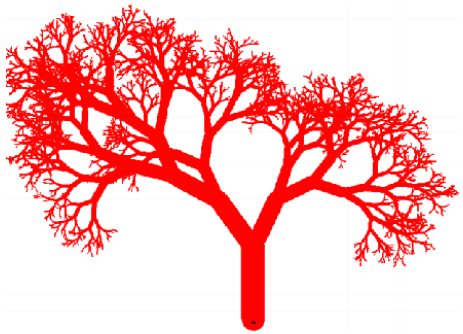
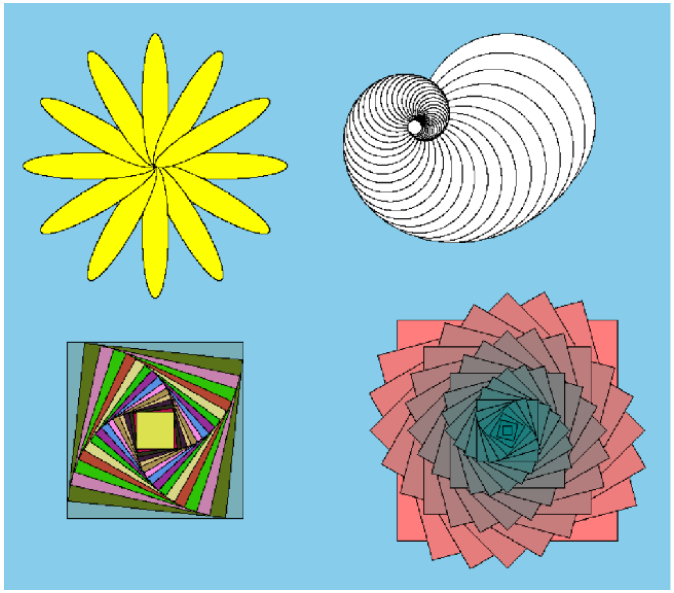


Image Source: (http://cs111.wellesley.edu/spring19)

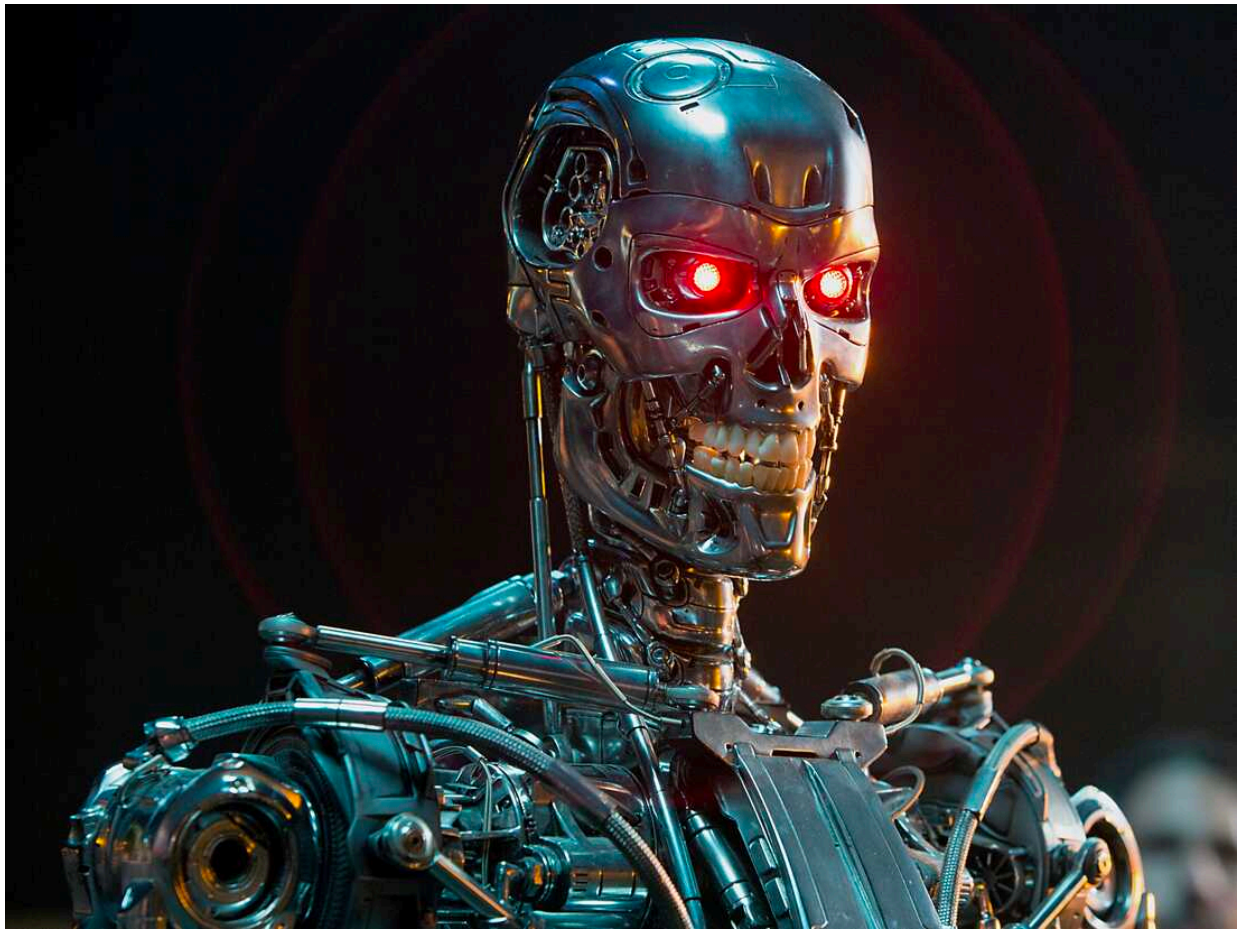


Beyond CS134

- For those interested in continuing on the CS path:
  - Obvious next step: take **CS136** + **Math 200**
  - Practice more Java over summer break: redo our labs in Java!
- In general, if you enjoy **puzzles and programming**, there are many ways to practice these skills:
  - Try Project Euler: Math + CS puzzles
  - MIT course: The missing semester of your CS eduction
- Staying connected with CS as non-majors:
  - Can still take CS136 and other courses!
  - Come talk to us for more ideas

Beyond CS134

- Now that you know all this stuff — what's next?
- Python for world domination?? Building killer robots is probably not the best use of your skills..



Beyond CS134

- Or you could get really creative with what you've learned.
- Tinker! It's not particularly essential that you always work on something "important", it could just be something that is "interesting" (even if only to you!)
- Many of us fall in love with computer science because it's a way for us to express our creativity — whether that's by building databases, teaching computers to identify interesting patterns in the data (machine learning), or even writing algorithms for creating computer generated music!
- Computer science has the potential to intersect with almost any other field that might interest you — statistics, physics, biology, philosophy, music, art.
- Eventually, you might find an intersection you love and call it your own!

An Example of Tinkering

- This quote from Joi (an AI agent) in Blade Runner 2049 is particularly fascinating:

  "Mere data makes a man — A and C and T and G. The alphabet of you — all from four symbols. I am only two — 1 and 0."

- Our DNA really is not so much more complicated (at least from a syntactic viewpoint) than the 1s and 0s that define the on and off behavior of circuits on our computer. Yet it produces such complex behavior!

- What do we (or rather some segments of our DNA) sound like to a computer? Well let's find out by generating some tones!

Course Evals Logistics

- Two parts: **(1) SCS form** , **(2) Blue sheets** (both online)
- Your feedback helps us improve the course for other students taking it in the future, and helps us shape the CS curriculum
    - Your responses are **confidential** and we will only receive a report of your anonymized comments after we have submitted all grades for this course
- **SCS forms** are used for tenure/promotion & seen by CAP etc, **blue sheets are open-ended** comments directed only to your instructor

*To access the online evaluations, log into **Glow** (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "**Course Evaluations**." Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can ask a neighbor for help or reach out to ir@williams.edu.*

Thank you!

- We made it!
- You all should be proud of how much you've learned
- Evals: We are always looking to improve the course and appreciate your constructive feedback
- **Thank you** for your patience and enthusiasm during these somewhat crazy times
- Good luck on finals and have a great summer!