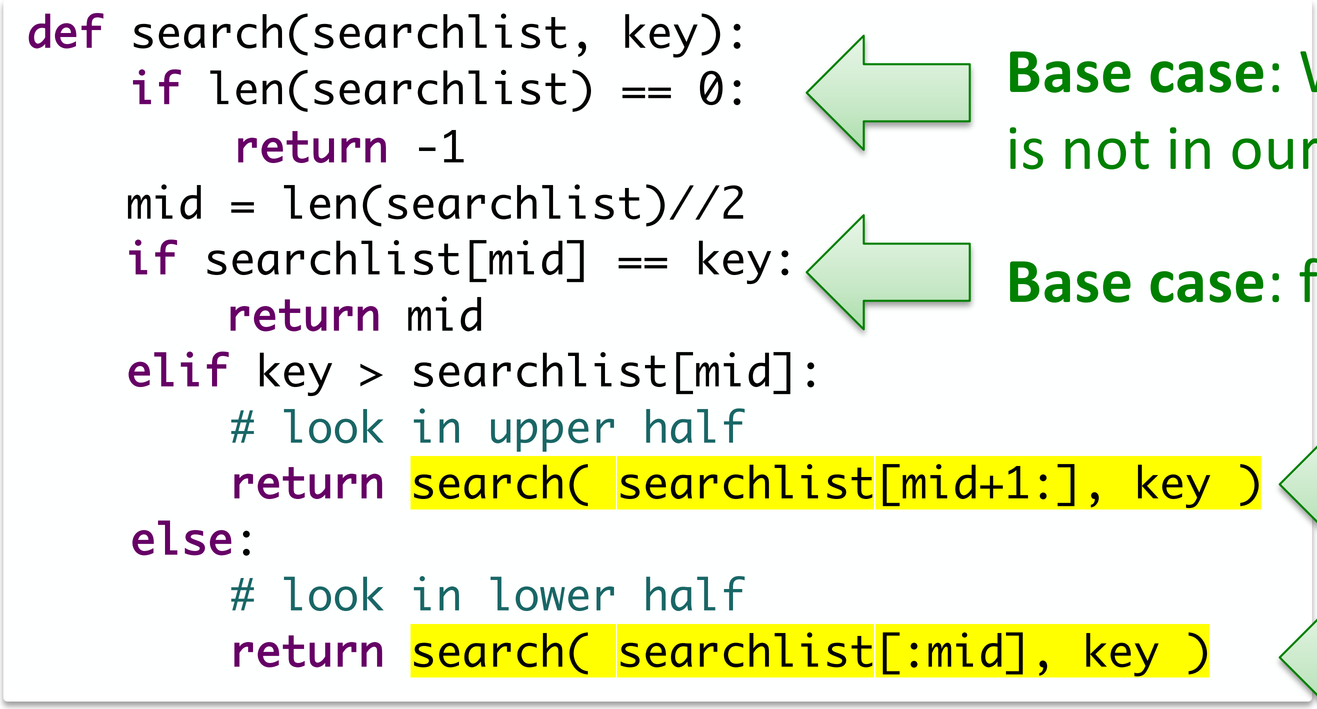
Objectives

- Computer Science is Complexity Science
- Course logistics
- BI: TikTok/Data
- Review for Final

Review

- What is recursion?
  - Provide an example of solving a problem recursively
- Programming languages
  - What are characteristics of programming languages?
  - What are common constructs in programming languages?
  - What are some differences between programming languages?

Review: Recursive Binary Search



```python
def search(searchlist, key):
    if len(searchlist) == 0:
        return -1
    mid = len(searchlist)//2
    if searchlist[mid] == key:
        return mid
    elif key > searchlist[mid]:
        # look in upper half
        return search( searchlist[mid+1:], key )
    else:
        # look in lower half
        return search( searchlist[:mid], key )
```

Base case: We know the key is not in our list

Base case: found it!

Recursion

Subproblem of *same* problem

Review: Recursion Summary

- ***Recursion***: method of solving problems
  - Break a problem down into smaller subproblems of the same problem until problem is small enough that it can be solved trivially
- Binary Search:
  - Break problem to ~half the size of original problem
  - Base cases: when the middle element is what you're looking for; when there are no elements in your list
- Any recursive problem can be solved iteratively
  - Some problems lend themselves better to recursive solutions

Review: Programming Language Characteristics

- **Syntax**: symbols used
- **Semantics**: what the symbols *mean*

Review: What is Computer Science?

"Computer Science is no more about computers than astronomy is about telescopes."

--Edsger Dijkstra

> A human must turn information into intelligence or knowledge. We've tended to forget that
> **no computer will ever ask a new question.**
>
> -- Grace Hopper

> Computers are incredibly fast, accurate, and stupid.
> Human beings are incredibly slow, inaccurate, and brilliant.
> Together they are powerful beyond imagination.
>
> -- Albert Einstein

Review: What This Course Is About

Problem Solving!

From *30 Rock*

Review: Parts of an Algorithm

- Input, Output
- Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

An overview for the semester!

COMPLEXITY SCIENCE

CS == Complexity Science

- How can it be done?
  - Based on **information**
  - Managing, manipulating data
  - Possible algorithms
- How well can it be done?
  - Most **efficient** algorithm in terms of time and/or space
- Can it be done at all?
  - Often, proof is a program--an implementation of the above

Computer Science != Programming

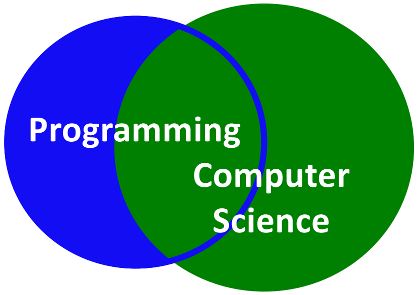
programming : CS ::

machining : engineering

grammar : literature

equations : mathematics

walking : W&L



a vehicle, not a destination

Computer Science Fields

Systems

- Architecture
- Operating systems
- Networks
- Distributed and parallel systems
- Databases
- Security
- …

Software

- Compilers
- Graphics
- Software engineering
- Software testing and verification
- …

Theory

- Algorithms
- Theory of computation
- …

Other

- Artificial intelligence
- Robotics
- Natural language processing
- Bioinformatics
- Visualization
- Numerical analysis
- …

- Often research involves combinations of these fields
- Not just programming!
  - But programming is a tool to do much, much more!

Computer Science Fields

Systems

- Architecture *
- Operating systems *
- Networks *
- Distributed * and parallel systems
- Databases
- Security
- …

Software

- Compilers
- Graphics *
- Software engineering*
- Software testing* and verification
- …

Theory

- Algorithms *
- Theory of computation
- …

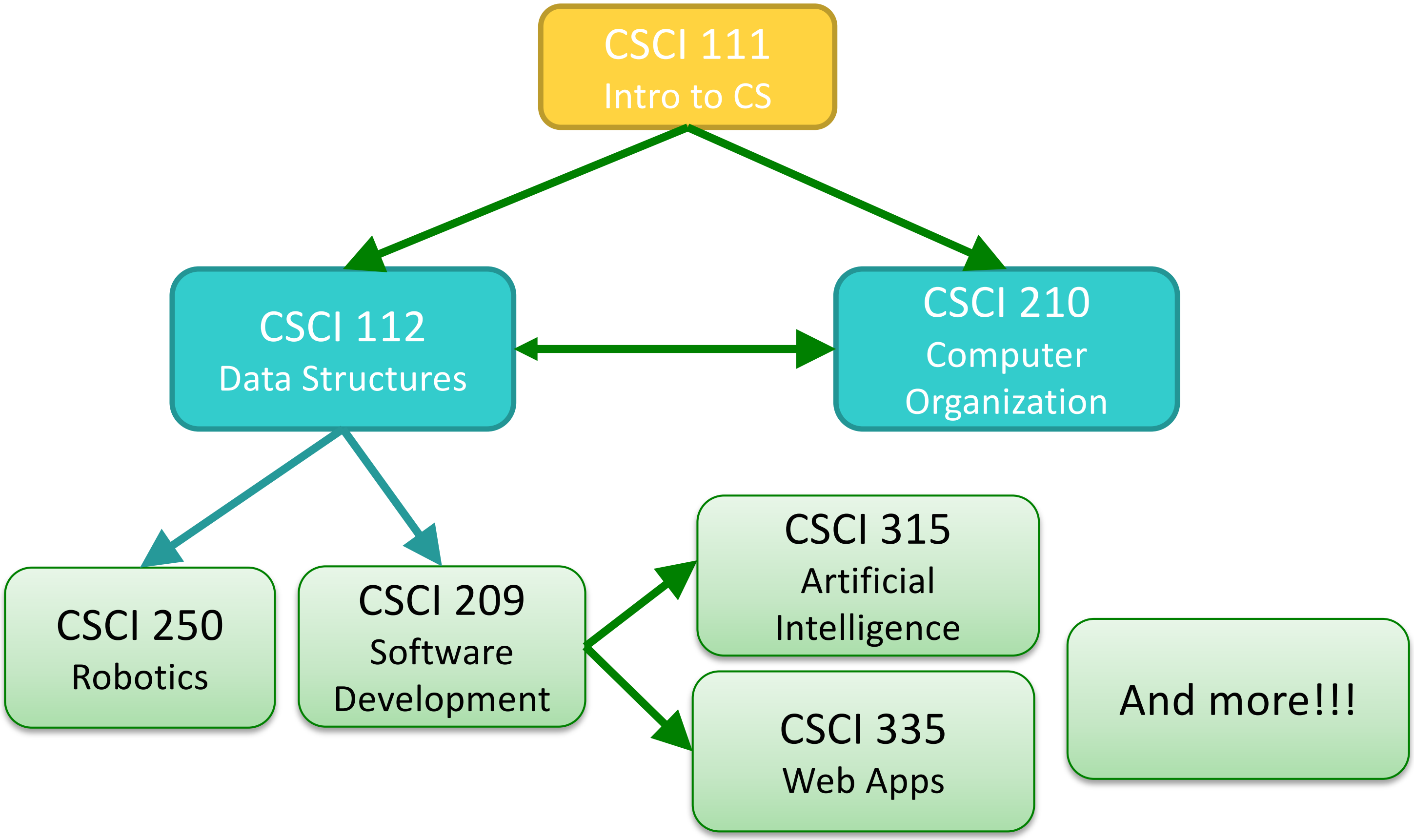
Other

- Artificial intelligence *
- Robotics *
- Natural language processing *
- Bioinformatics
- Visualization*
- Numerical analysis
- …

* = field we discussed or did a problem in

- Some are a stretch :)

Where Can You Go from Here?



Course Conclusions

- Better [computational] problem solver
- See impact of computer science on your life
    - Think differently about issues
- Understand some computing issues better
    - Taking out some of the mystery
    - Testing, debugging, efficiency
- Algorithms are everywhere
    - Process for solving problems, **efficiently**
    - Mapping human intuition to systematic/automatic process

Final Exam

- Timed exam on Canvas
  - Some questions “in” Canvas
  - Some questions in a Word document
- Only open brain, Canvas, Word
- Closed everything else
  - Turn off notifications, hide distractions
- Can have paper for scratchwork

Final: Word Part

- One question in Canvas has the Word document
- Download document, type your answers in document
  - I only left a few lines between questions
  - Write your answer below/between the questions
  - Use the point amount to help gauge how much to write
  - Be careful about autocorrect (e.g., avoid `i` as a variable)
- Submit/upload Word document

Final Exam Content

- Focus on object-oriented programming
- New content: search techniques, lists (1D and 2D), programming languages, recursion, complexity science
- Cumulative:
  - Functions, data types, common methods & operations
  - How to model data

Your questions?

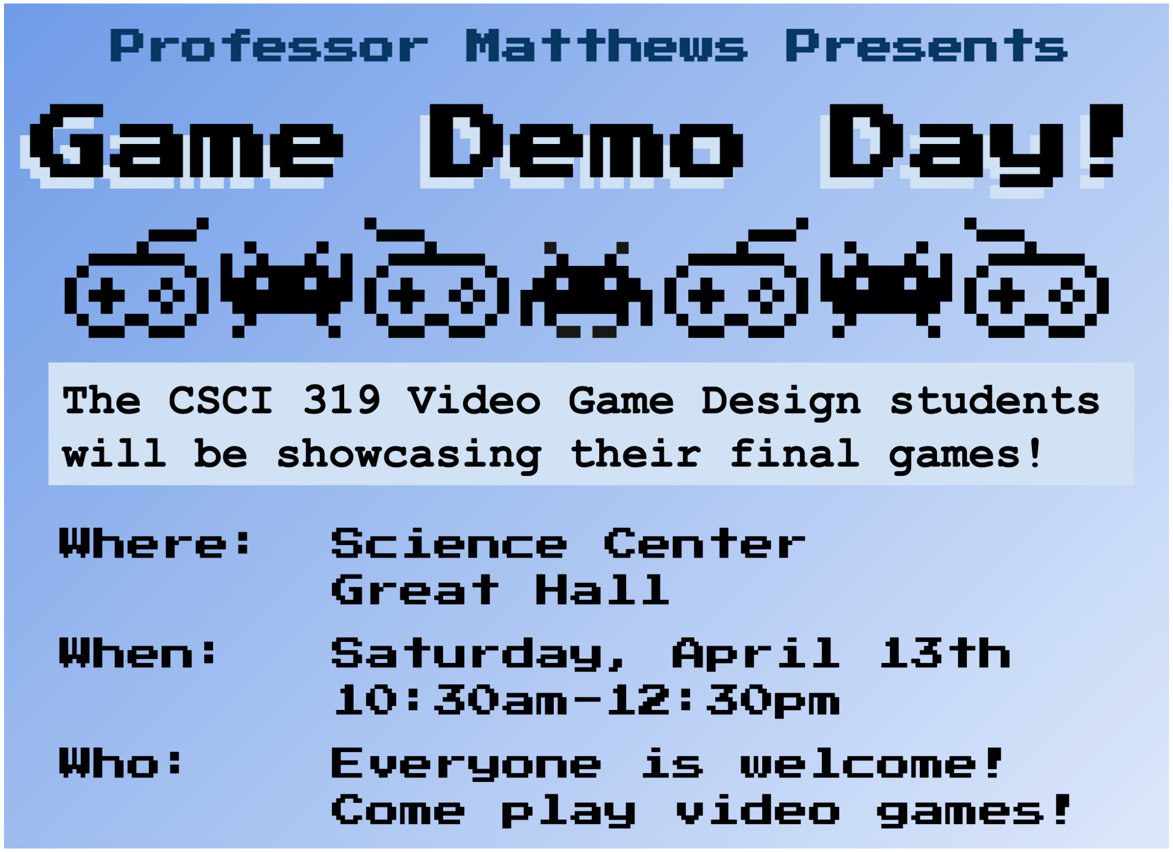
Course Evaluations

- On Canvas, due Monday
- Incentive
  - If 60% of students complete evaluation, 1% Extra Credit on *lab* grades
  - For each additional 10% of students who complete evaluation, 1% additional EC on *lab* grades
  - Total possible EC: 5%

Looking Ahead

- Deadline: Monday 11:59 p.m.
  - Course evaluations
  - All (late) lab work
- Deadline: tonight at 11:59 p.m.
  - Extra credit articles reviewed
  - Spend time studying for final exam (worth more)
- Deadline: Friday at noon: Final Exam due
- Now: Broader Issue Discussion
- Next: Final Exam Review

Extra Credit Opportunity



Professor Matthews Presents

Game Demo Day!

The CSCI 319 Video Game Design students will be showcasing their final games!

Where: Science Center
Great Hall

When: Saturday, April 13th
10:30am-12:30pm

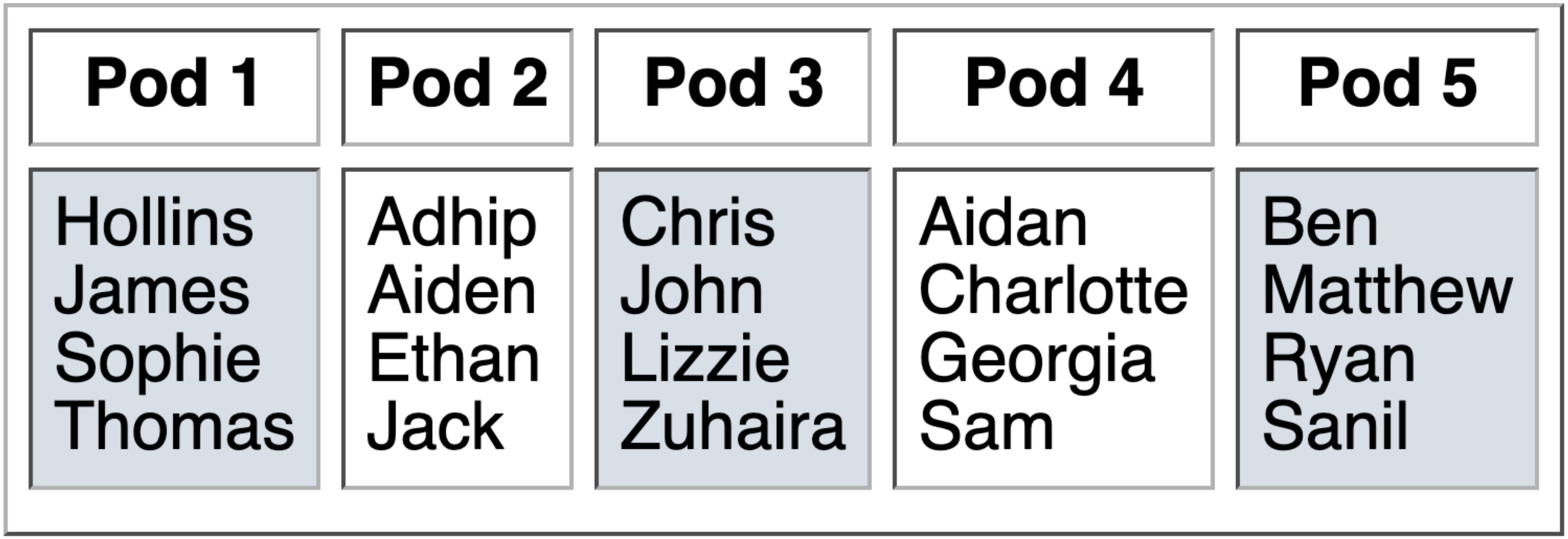
Who: Everyone is welcome!
Come play video games!

Evaluate up to 2 games on Canvas for up to 10 points Extra Credit towards labs

BROADER ISSUE: TIKTOK/DATA

https://www.eff.org/deeplinks/2024/03/5-big-unanswered-questions-about-tiktok-bill

Broader Issue Groups



Pod 1	Pod 2	Pod 3	Pod 4	Pod 5
Hollins James Sophie Thomas	Adhip Aiden Ethan Jack	Chris John Lizzie Zuhaira	Aidan Charlotte Georgia Sam	Ben Matthew Ryan Sanil

Broader Issue: TikTok/Data

- What problems is the legislation trying to solve?
- What will the impact of the legislation be?
  - Why does *this* bill have bipartisan support?
- What do you think the problems are?
  - Does the legislation solve those problems?
- From student: Who gets to dictate what data privacy looks like? The user? The government? The company?

- What are your takeaways?

Make Good Decisions!

Final Exam Review

- What is our process for developing classes?
- What are the different ways to iterate through a list?
- How do you iterate through a dictionary?

Animal Shelter Software

- We want to keep track of animals at an animal shelter

What is our process for developing a class?

Process

- Determine data, functionality
- Create class
  - Create __init__, __str__ methods
- Test
- Create additional methods, test

Class: Pet

- Data:
  - Species of animal (dog, cat, chinchilla)
  - Name
    - Defaults to “”
  - Status (in holding, in adoption room, adopted)
    - Defaults to “in holding”
- Functionality
  - Constructor: `Pet(species)`
  - String format: `"species: name, status"`
  - Setters for name
  - Set animal as adopted or in adoption room
  - Getters for this information

Counter Class Specification

- Implement, Test
- Example use: Caesar cipher

- A class that represents a counter that wraps around from a high value back to its low value
- Data:
  - Low, high, and current values (all integers)
- Functionality:
  - Constructor – takes as parameters the low value and the high value
    - counter starts at low value
  - A string representation of the Counter
    - Format: “low: <low> high: <high> current: <current>”
  - Getters: low, high, current value
  - Increment the counter by a given amount (a positive amount), wrapping around to low again, if necessary. Returns number of times had to wrap around.
    - Example: if counter’s low is 0 and the high is 9 and its current value is 9:
      - `test.testEqual(counter.increment(1), 1); test.testEqual(counter.getCurrent(), 0)`
  - Decrement the counter by a given amount (a positive number), wrapping around to high again, if necessary. Returns number of times had to wrap around.
  - Sets the counter's value, only if low <= value <= high. Otherwise, prints an error message.

Palindrome

- Write a program that determines if a string (input by a user) is a palindrome. A *palindrome* is a word that is the same forwards and backwards. Some example palindromes: "kayak", "A man A plan A canal Panama".
- http://www.fun-with-words.com/palin_example.html
- Break the problem into at least two functions:
  - main
  - isPalindrome, which returns True iff the parameter string passed into the function is a palindrome.
- Depending on how you think about the problem, you may want to break the solution into more functions, e.g., a reverseString function

Generate a Random Password

- Function: given number of characters
- Returns a random password
  - Includes upper, lowercase letters; numbers; punctuation

Function: createDict

- Write a function that, given two lists of equal length
  - The first list is the keys
  - The second list is the values
- Returns the dictionary that maps the keys from the first list to the values in the second list, respectively/in order
- Examples:
  - test.testEqual(createDict([1, 2], [“one”, “two”]), {1:”one”, 2:”two”})
  - test.testEqual(createDict([1, 2], [“two”, “one”]), {1:”two”, 2:”one”})

Fibonacci

- Solve the Fibonacci sequence *recursively*
- Note: this is less efficient than the iterative solutions you wrote during lab