Overview
- Recursion
  - Demo
    - View example & analysis
    - Demonstration
  - You already know it
  - Definition
  - Trust the Recursion!
  - Conclusion

Definition
- Recursion: {noun} See recursion. 😊
- An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task
- Recursive solutions involve two major parts:
  - Base cases: the problem is simple enough to be solved directly
  - Recursive cases: A recursive case has three components
    - Divide the problem into one or more smaller or similar parts
    - Invoke the function recursively on each part, and
    - Combine the solutions of the parts into a solution for the problem.
- Depending on the problem, any of these may be trivial or complex.

You already know it!

Trust the Recursion
- When authoring recursive code:
  - The base is usually easy: "when to stop?"
  - In the recursive step
    - How can we break the problem down into two:
      - A piece I can handle right now
      - The answer from a smaller piece of the problem
    - Assume your self-call does the right thing
    - How to combine parts to get the overall answer?
  - Practice will make it easier to see idea

Summary
- Behind Abstraction, Recursion is probably the 2nd biggest idea about programming in this course
- It's tremendously useful when the problem is self-similar
- It's no more powerful than iteration, but often leads to more concise & better code