Michael Ball

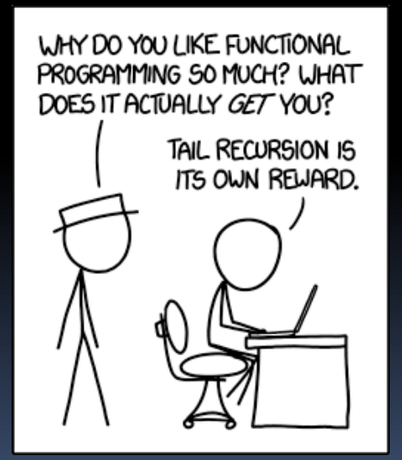
The Beauty and Joy of Computing

Lecture #2
Functions

Chip Design Savings from Functional Language?

A new functional language (you'll learn what that means today) *"makes it possible a chip design transformation is completely error-free"*. Huge potential cost-savings!
"Functional Programming is not part of the standard curriculum".
It is alive and well in Snap*!*, yay!

(xkcd.com/1270)



www.utwente.nl/en/newsevents/!/2015/1/167544/massive-chip-design-savings-to-be-realized

When Do You Learn Things in CS10?

- **Lecture**
  - Computing in the News + Discussion
  - Big ideas, Inspiring Introductions, Demos
  - **NOT THE CODING DETAILS**
- **Lab, Homework, Projects**
  - Coding, Collaboration, Deep Learning
- **Reading**
  - Context, Impact of Computing, Current Events
- **Discussion**
  - Clarify week's material, Unplugged Activities

Office Hours and Discussions

- You can go to as many office hours and discussions as you wish!
  - You're not just limited to your TA
- Please check the schedule on cs10.org, as that will have the currently correct times.
- My OH – right after lecture on Tues (5-6pm), before lecture on Thurs (3-4pm)

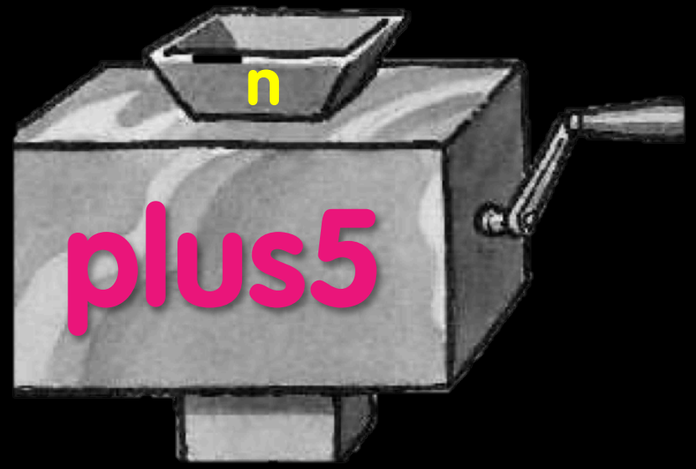
Function Basics

Abstraction: Generalization REVIEW

- You are going to learn to write functions, like in math class:

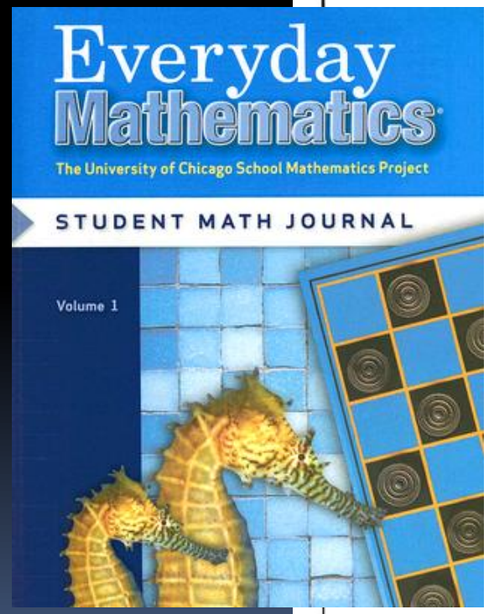
$$y = plus5(n)$$

  - plus5 is the function
  - n is the input, a number
  - It returns a single value, here a number 5 more than the input.



Function machine
(*Simply Scheme,* Harvey)

Functions in 2nd Grade Math Curricula!



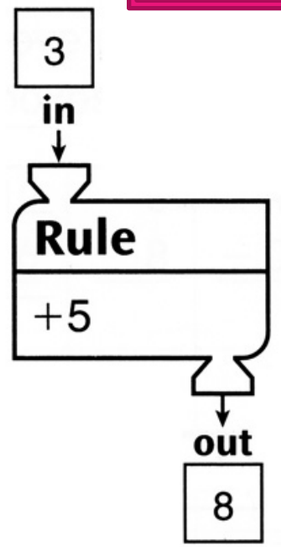
HOME LINK 2·11

"What's My Rule?"

Family Note Today your child learned about a kind of problem you may not have seen before. We call it "What's My Rule?" Please ask your child to explain it to you.

Here is a little background information: Imagine a machine with a funnel at the top and a tube coming out of the bottom. The machine can be programmed so that if a number is dropped into the funnel, the machine does something to the number, and a new number comes out of the tube. For example, the machine could be programmed to add 5 to any number that is dropped in. If you put in 3, 8 would come out. If you put in 7, 12 would come out.

We call this device a *function machine.*

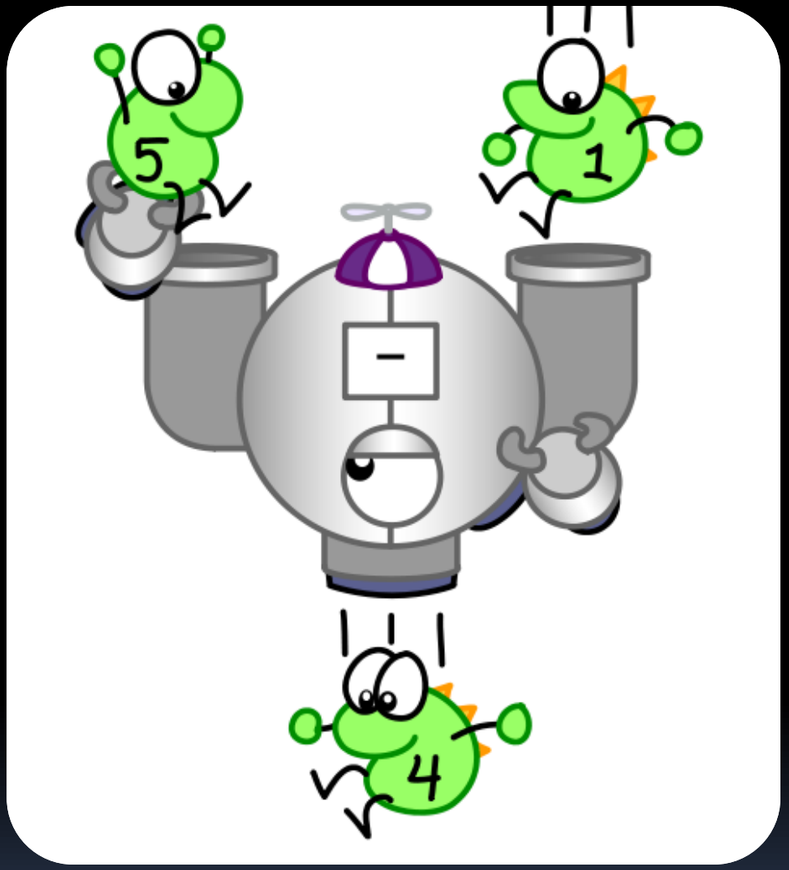


You can show the results of the rule "+5" in a table:

in	out
3	8
7	12
15	20

Function Definition

- **Functions take in 0 or more inputs, return exactly 1 output**
- **The same inputs MUST yield same outputs.**
  - Output function of input only
- **Other rules of functions**
  - No state (prior history)
  - No mutation (no variables get modified)
  - No side effects (nothing else happens)



Function Metaphor
(*CS Illustrated,* Ketrina Yim)

Which is NOT a function?

a) pick random ( ) to ( )

b) ( ) < ( )

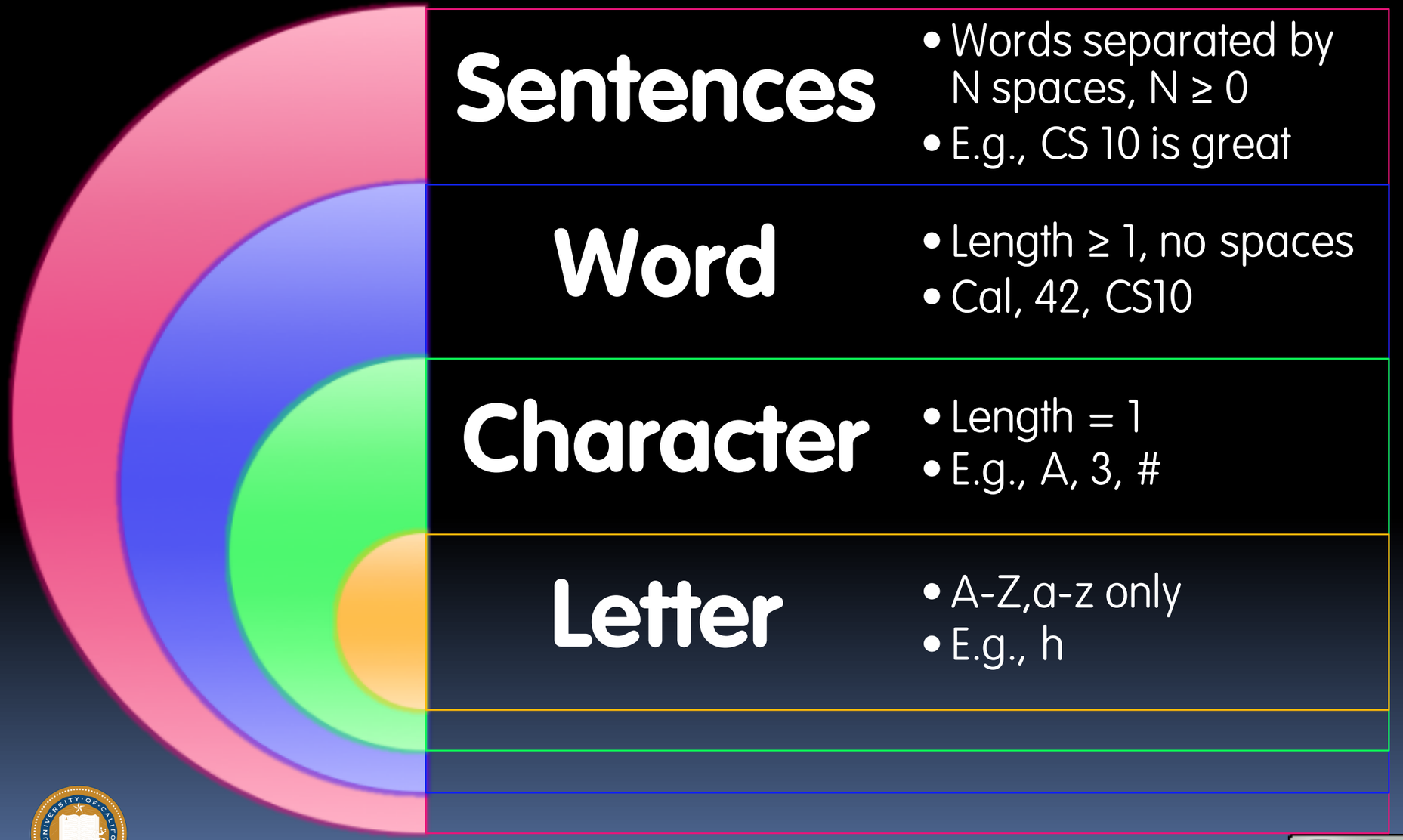
c) length of ( )

d) sqrt of ( )

e) true

Data Types
Domain & Range

Basic Data Types (You'll make more)



Domain and Range (from Math)

Domain

- The "class" of input a function accepts
- **Examples**
  - Sqrt of
    - Non-negative numbers
  - Length of
    - Sentence, word, number
  - _ < _
    - Sentence, word, number
  - _ and _
    - Boolean

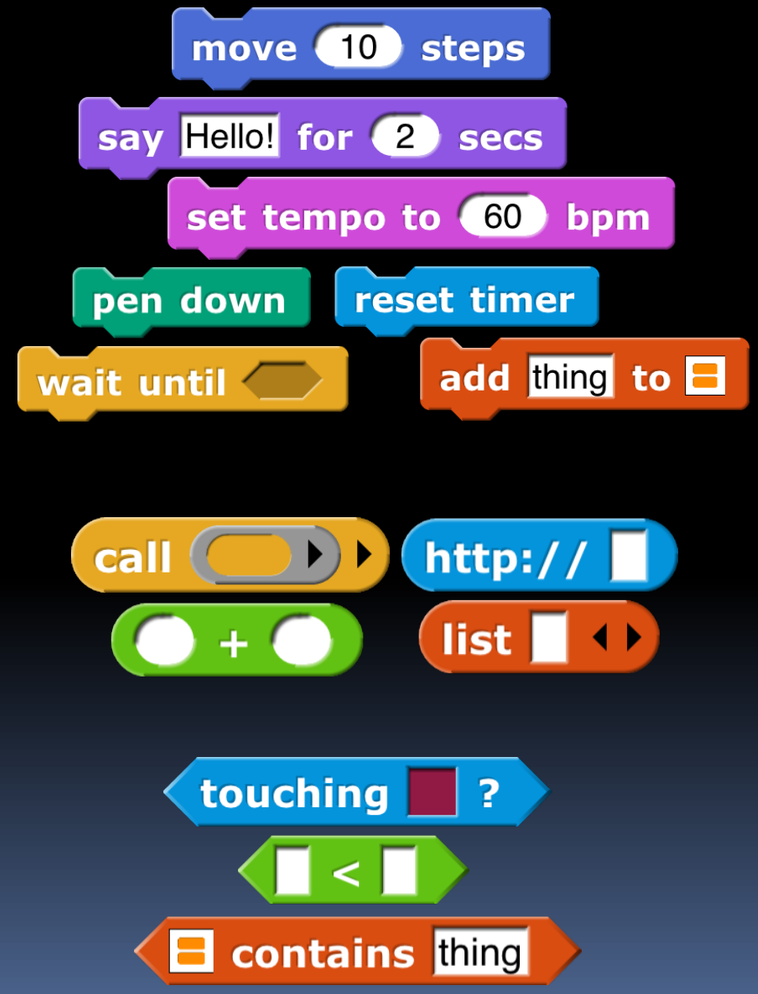
Range

- All the possible return values of a function
- **Examples**
  - Sqrt of
    - Non-negative numbers
  - Length of
    - Non-negative integer
  - _ < _
    - Boolean (true or false)
  - _ and _
    - Boolean (true or false)

Types of Blocks in Snap!

Procedures, Subroutines

- **Command**
  - No outputs, meant for side-effects
  - Not a function…
- **Reporter (Function)**
  - Any type of output
- **Predicate (Function)**
  - Boolean output
    - (true or false)

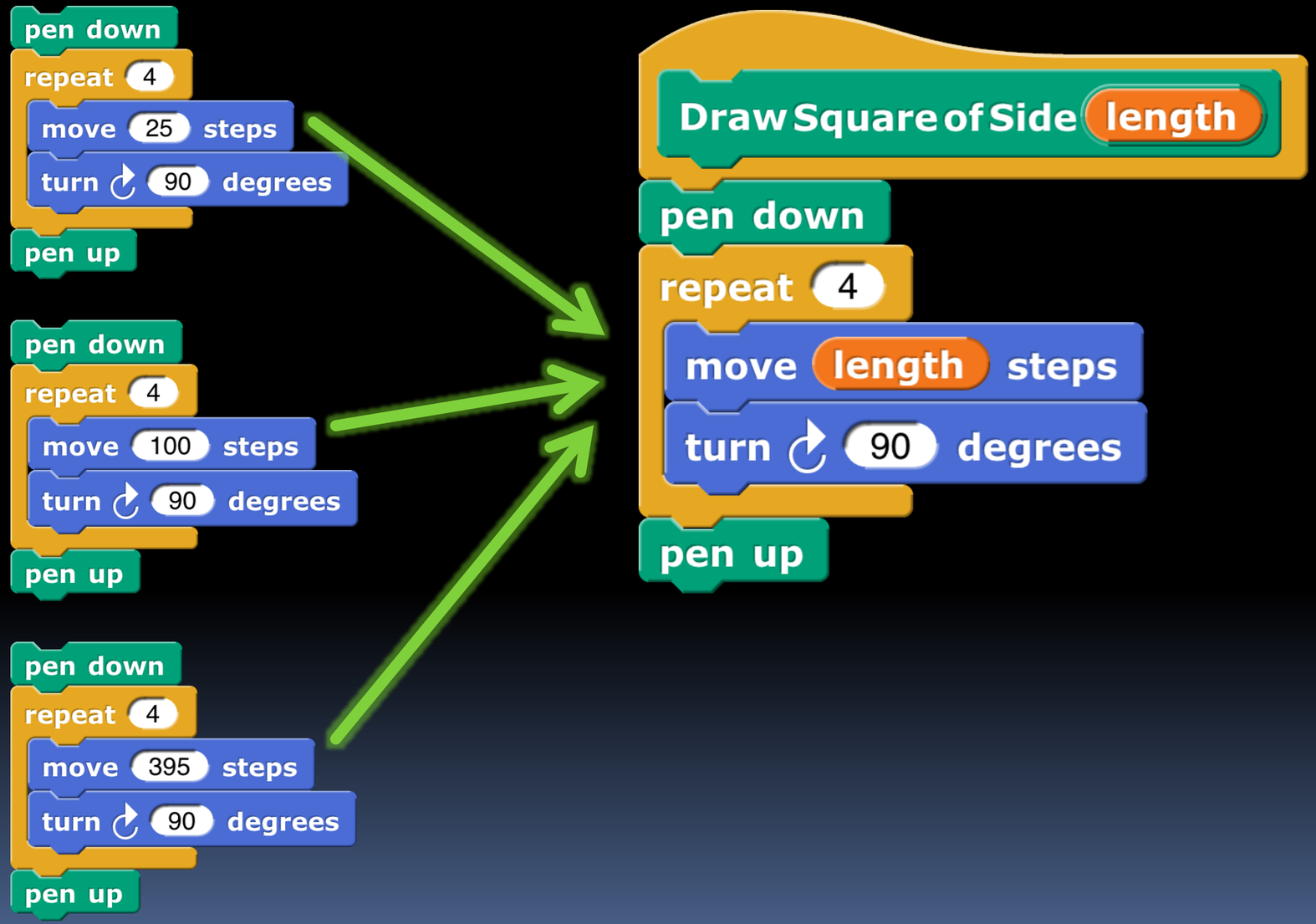


Domain, Range of…

Domain	Range
a) Integer ≥ 1, Number	Digit
b) Integer ≥ 1, Word	Letter
c) Integer ≥ 1, Word	Character
d) Integer ≥ 1, Sentence	Letter
e) Integer ≥ 1, Sentence	Character

Why Should You Use Functions?

Why Use Functions? (1/3)



Why Use Functions? (2/3)

- They allow for **generalization** of code!
- The **building blocks** of our programs
- They can be **composed together** to make even more magnificent things.
- Breaking big problems down into smaller ones is **functional decomposition**

Why Use Functions? (3/3)

- **If a function only depends on the information it gets as input, then nothing else can affect the output.**
  - It can run on any computer and get the same answer.
- **This makes it easy to parallelize computation.**
  - Functional programming is a great model for writing software that runs on multiple systems at the same time.

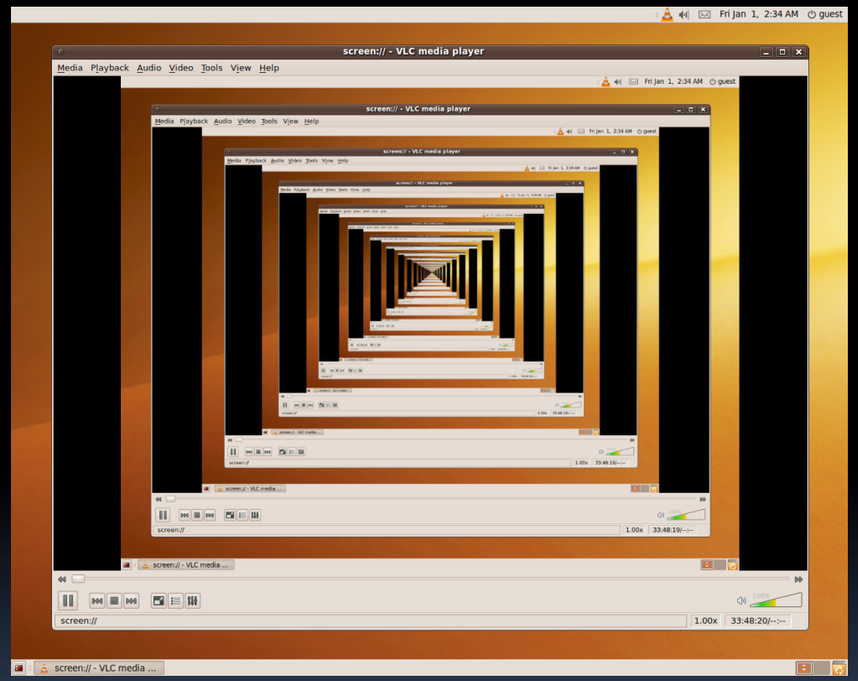
Cabinet Aisle in a Datacenter
(Wikipedia, Robert Harker)

Quick Preview: Recursion

Recursion is a technique for defining functions that use **themselves** to complete their own definition.

This is one of our Big Ideas!

Recursion in Screen Recording Program
(*Wikipedia,* Hidro)



Ball

Functions Demo!

Functions Summary

- **Abstraction: Generalization**
- **Computation is the evaluation of functions**
  - Plugging pipes together
  - Function: ≥ 0 inputs, 1 output
  - Functions can be input!
- **Features**
  - No state
    - E.g., variable assignments
  - No mutation
    - E.g., changing variable values
  - No side effects
    - E.g., nothing else happens

$$f(x) = (x+3) * \sqrt{x}$$