UC Berkeley EECS
Sr Lecturer SOE
Dan Garcia

The Beauty and Joy of Computing

Lecture #4 : Functions

Quest (first exam) in **in 14 days!!**

THE FUTURE OF VIDEO GAMES?

Valve (video game makers of Half-Life) believes the future of video games may not be in the *input* device (ala the Wii remotes or your body ala Kinect), but the output device! What is shown on the right is an *augmented reality* device, layering 3D content onto the real world.

http://www.nytimes.com/2012/09/09/technology/valve-a-video-game-maker-with-few-rules.html

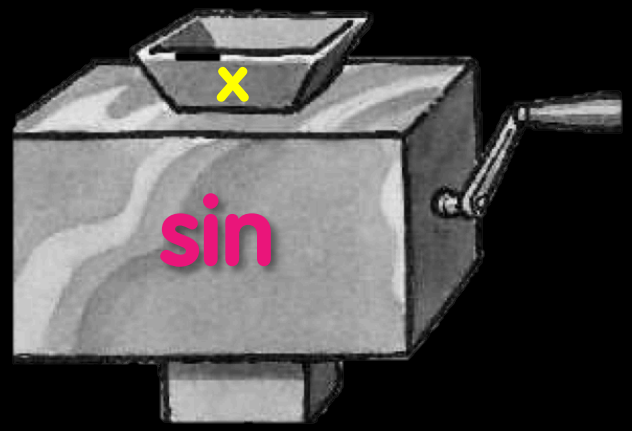
Enrollment Check-in (done live)

Generalization (in CS10) REVIEW

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

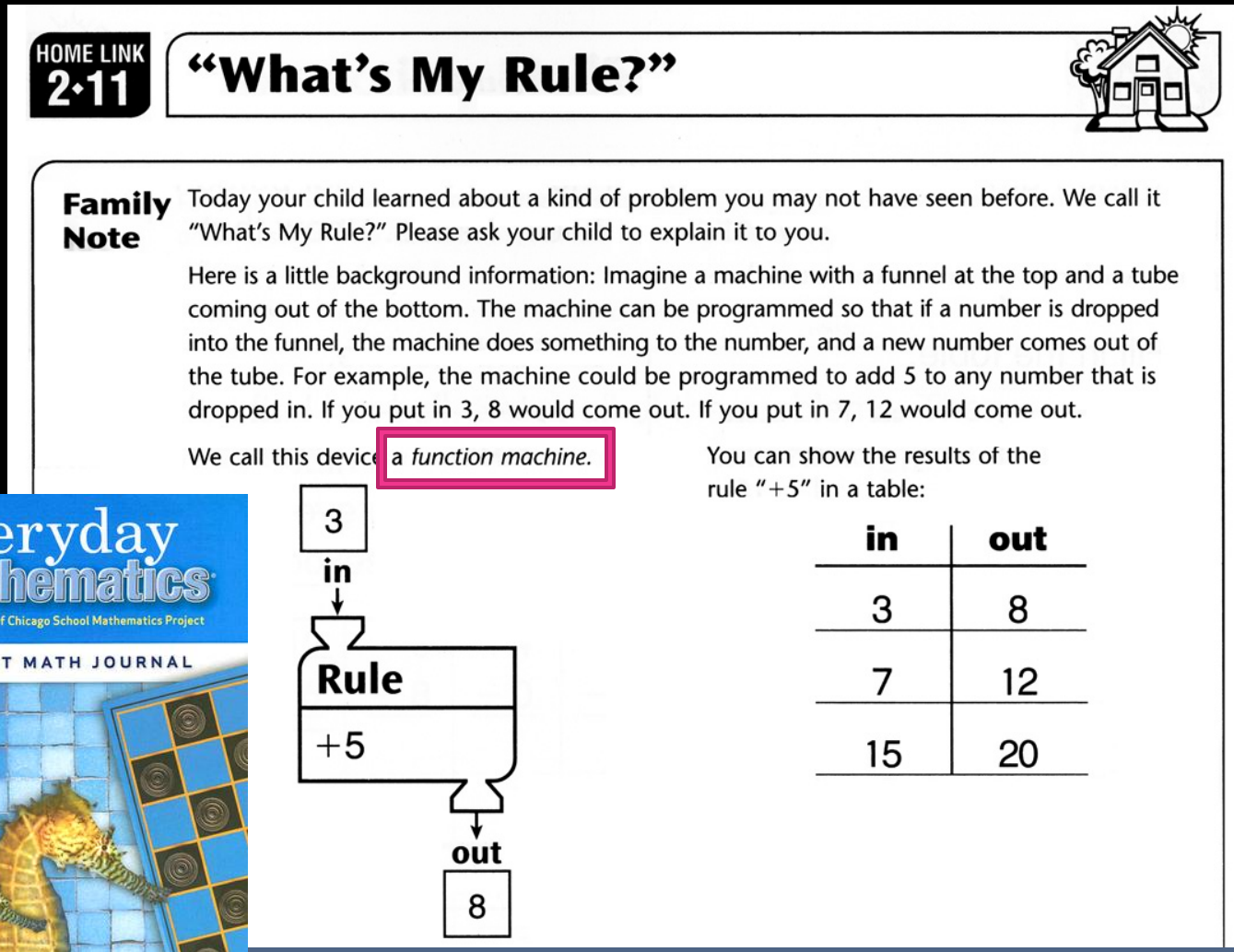
$$y = \sin(x)$$

  - sin is the function
  - x is the input
  - It returns a single value, a number



"Function machine" from *Simply Scheme* (Harvey)

Dan's kid's 2nd grade HW *yesterday!*



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 devic a *function machine.*

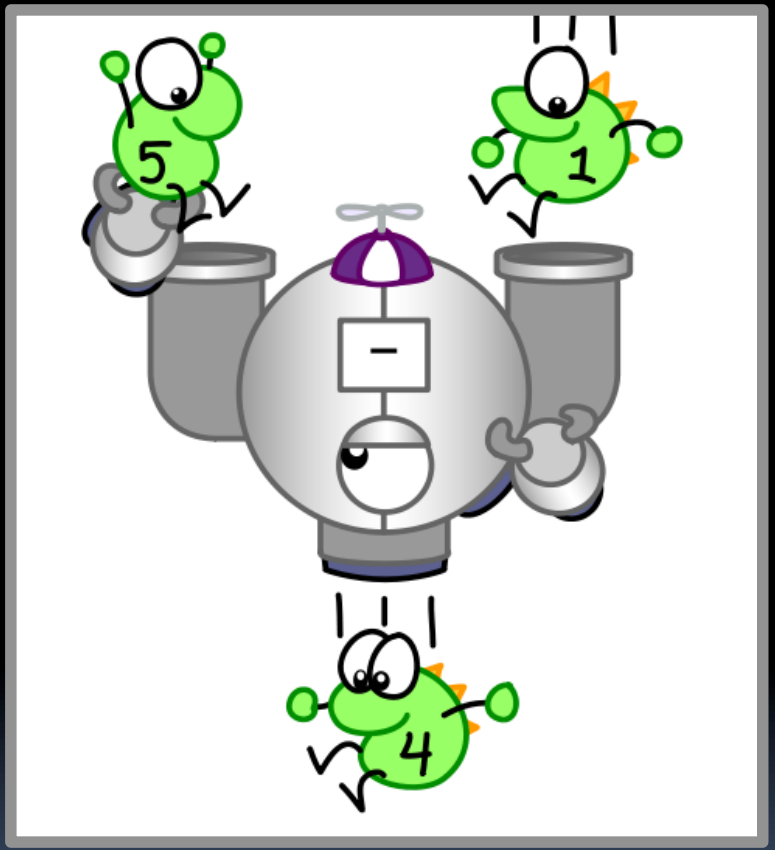
3
in
Rule
+5
out
8

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

in	out
3	8
7	12
15	20

Function basics

- Functions take in **0 or more inputs** and 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)



CS Illustrated function metaphor

Which is NOT a function?

a) pick random ( ) to ( )

b) ( ) < ( )

c) length of ( )

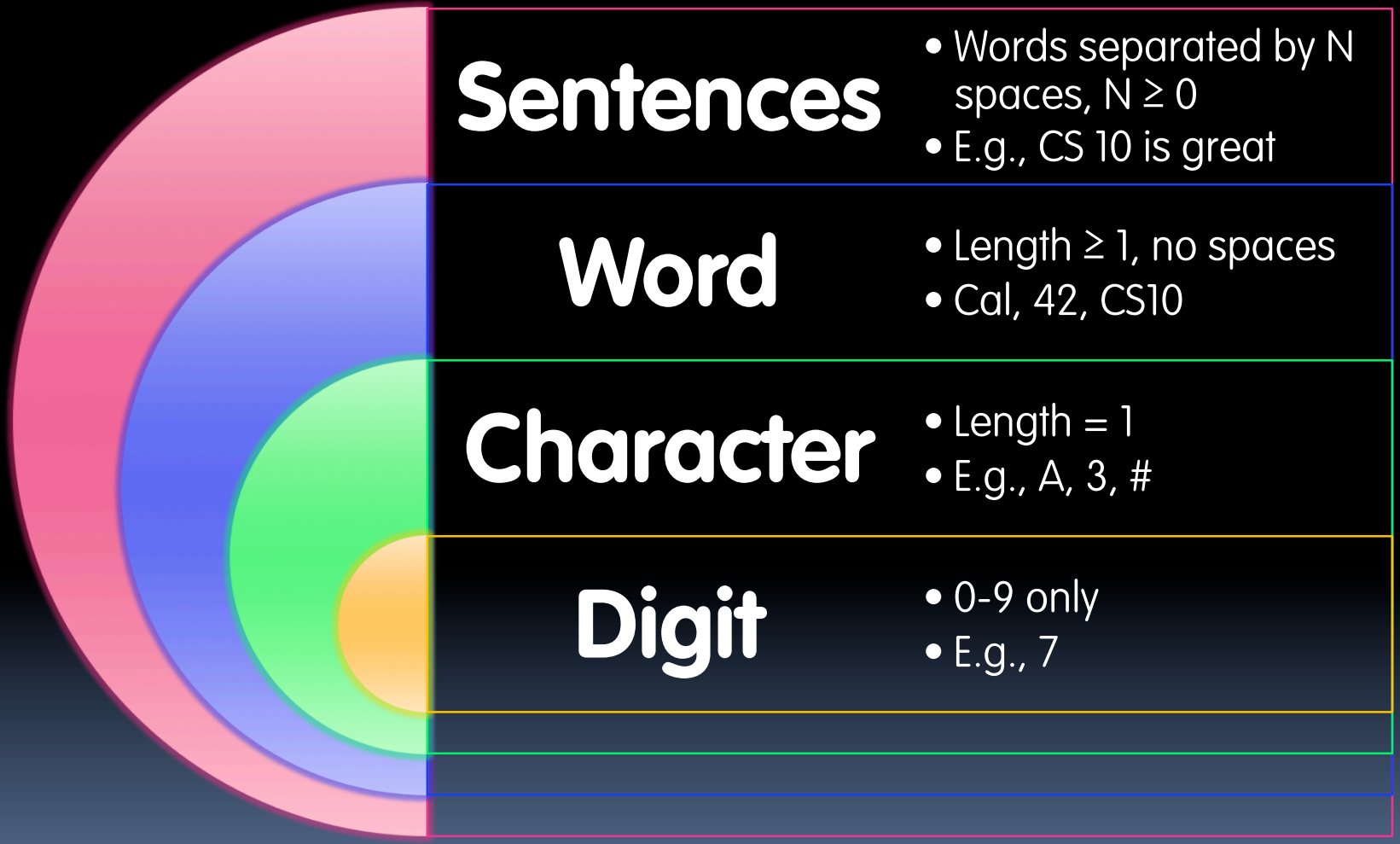
d) sqrt of ( )

e) true

More Terminology (from Math)

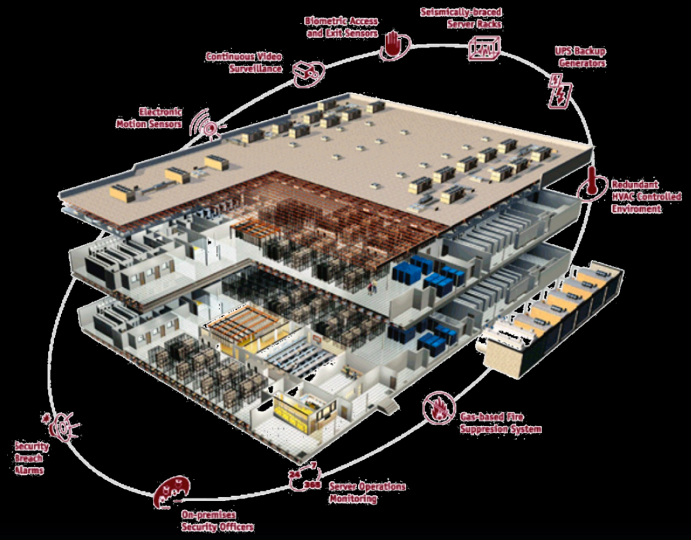
- **Domain**
  - The "class" of input a function accepts
- **Examples**
  - Sqrt of
    - Positive numbers
  - Length of
    - Sentence, word, number
  - _ < _
    - Both: Sentence, word, number
  - _ and _
    - Booleans
  - Letter _ of _
    - Number from 1 to input length
    - Sentence, word, number
- **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)
  - Letter _ of _
    - Letter

Types of input (there are more)



Why functions are great!

- **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 incredibly easy to parallelize functions.**
  - Functional programming is a great model for writing software that runs on multiple systems at the same time.



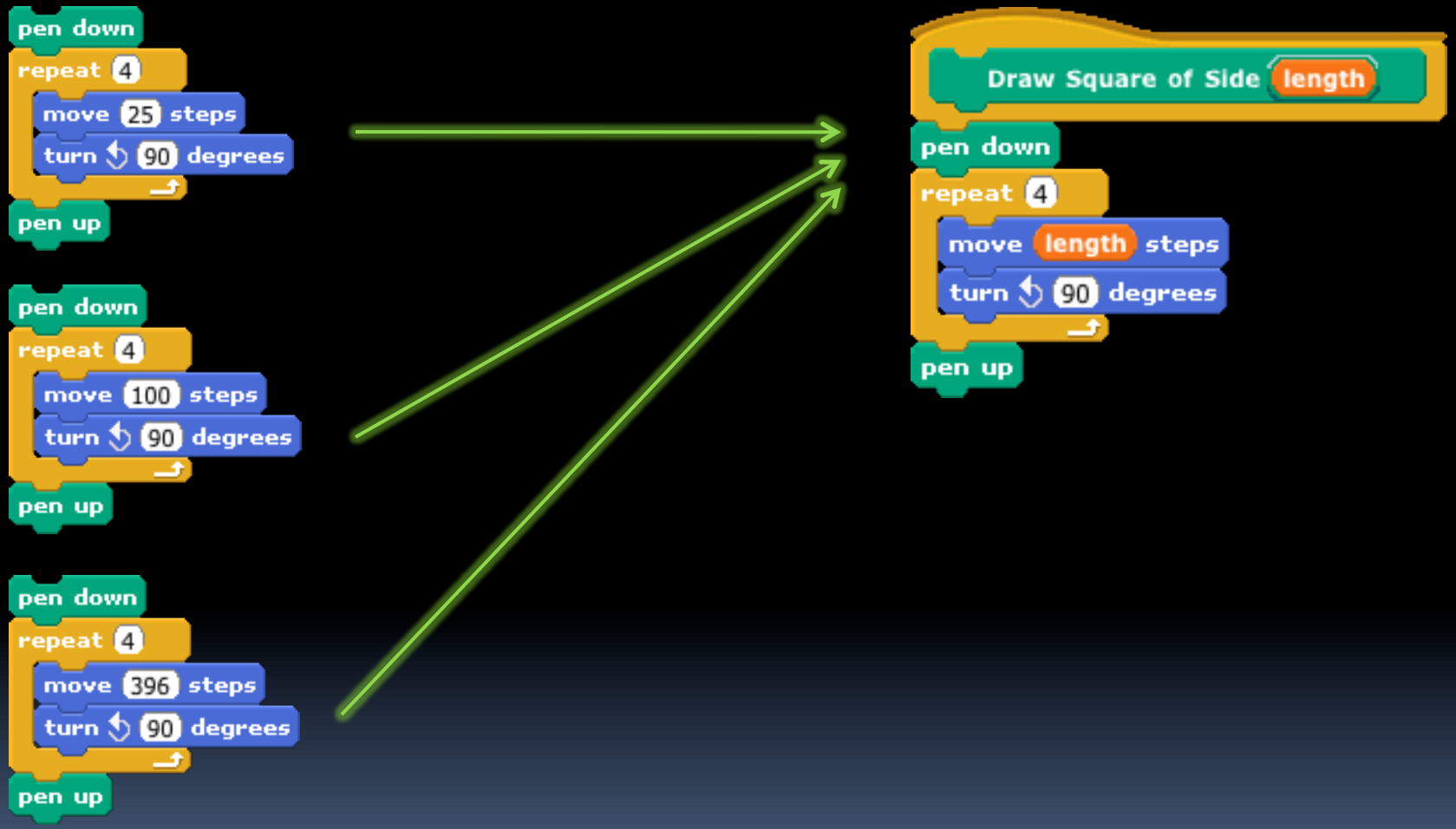
Datacenter

Scratch → BYOB (Build Your Own Blocks)

λSnap!

- **Scratch**
  - Invented @ MIT
  - Maintained by MIT
  - Huge community
  - Sharing via Website
  - No functions ☹
  - Scratch 2.0 in Flash
    - No iOS devices. ☹
  - scratch.mit.edu
- **BYOB (to be "Snap!")**
  - Based on Scratch code
  - Maintained by jens & Cal
  - Growing community
  - No sharing (yet) ☹
  - Functions! ☺ … "Blocks"
  - BYOB 4.0 in HTML5
    - All devices ☺
  - byob.berkeley.edu

Why use functions? (1)



The power of **generalization**!

Why use functions? (2)

They can be composed together to make even more magnificent things.

They are literally the building blocks of almost everything that we create when we program.

We call the process of breaking big problems down into smaller tasks functional decomposition

Types of Blocks

- **Command**
  - No outputs, meant for side-effects
  - Not a function…

- **Reporter (Function)**
  - Any type of output

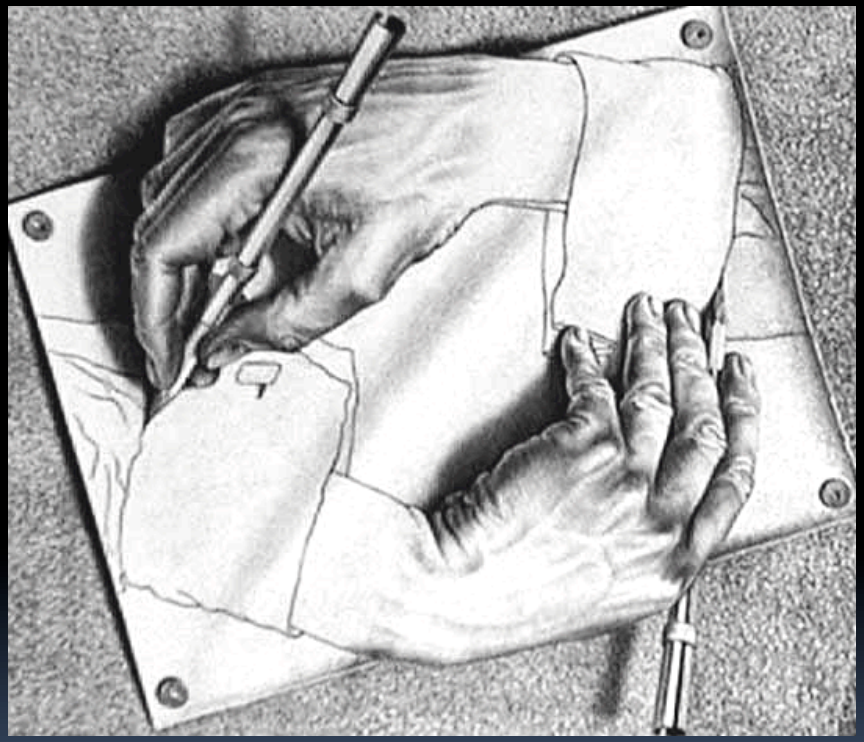
- **Predicate (Function)**
  - Boolean output
    - (true or false)

Garcia

Quick Preview: Recursion

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

M. C. Escher : *Drawing Hands*



We will spend a lot of time on this.

Functional Programming Summary

- **Computation is the evaluation of functions**
  - Plugging pipes together
  - Each pipe, or function, has exactly 1 output
  - Functions can be input!
- **Features**
  - No state
    - E.g., variable assignments
  - No mutation
    - E.g., changing variable values
  - No side effects
- **Need BYOB not Scratch**

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