UC Berkeley EECS
Lecturer SOE
Dan Garcia

CS10
The Beauty and Joy of Computing

Lecture #4 : Functions

2011-09-12

Quest (first exam) in this room **in 7 days!!**

KINECT? YOUR BODY IS ANTENNA!

Researchers at Microsoft and UW are working on a system that uses the fact that your body can act as an antenna and notes how ambient electric fields change to figure out what your position or motion was. The idea is you don't need a camera or Wiimote to interact with it!

`www.nytimes.com/2011/09/11/business/using-gestures-to-control-electronic-devices.html`

Generalization (in CS10) REVIEW

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

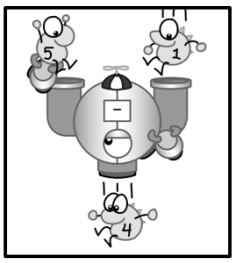
y = sin(x)

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

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

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)



Which is NOT a function?

a) pick random to

b) <

c) length of

d) sqrt of

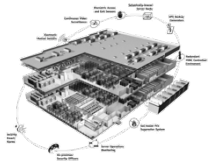
e) true

More Terminology (from Math)

- **Domain**
  - The "class" of input a function accepts
  - Examples
- **Range**
  - All the possible return values of a function
  - Examples

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.

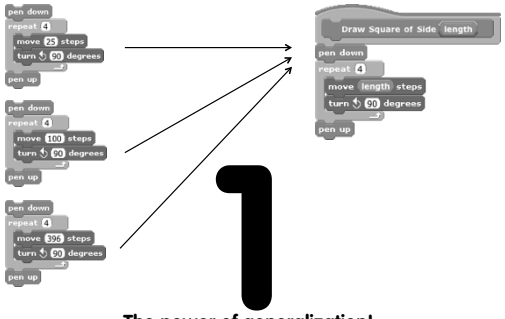


Datacenter

Scratch → BYOB (Build Your Own Blocks)

- **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

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

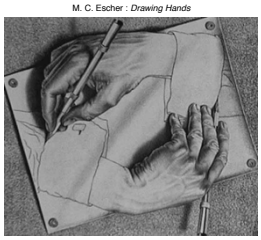
- **Predicate (Function)**
  - Boolean output
    - (true or false)

Quick Preview: Recursion

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

We will spend a lot of time on this.

M. C. Escher : *Drawing Hands*



en.wikipedia.org/wiki/Functional_programming

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**

$$f(x) = (2+3) * \sqrt{x}$$