Use this area for notes

CS10 News

UC Berkeley Teaching Professor Dan Garcia

The Beauty and Joy of Computing

Functions

The Resurgence of Functional Programming!

…functional programming allows you to 'solve problems, not solve puzzles,' and that can lead to increased focus, flow, and joy. – Gene Kim

If he could wave a magic wand, Kim would want everyone to learn a functional programming language, so they could experience the same joy he has found.

(xkcd.com/1270)

WHY DO YOU LIKE FUNCTIONAL PROGRAMMING SO MUCH? WHAT DOES IT ACTUALLY GET YOU?
TAIL RECURSION IS ITS OWN REWARD.

www.infoq.com/news/2020/11/functional-programming/

(Cal) 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

(Cal) 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's office hours
- Please check the schedule on cs10.org, as that will have the currently correct times.
  - Dan's OH: Fridays 1-2pm (same as lecture) in 777 Soda

Function Basics

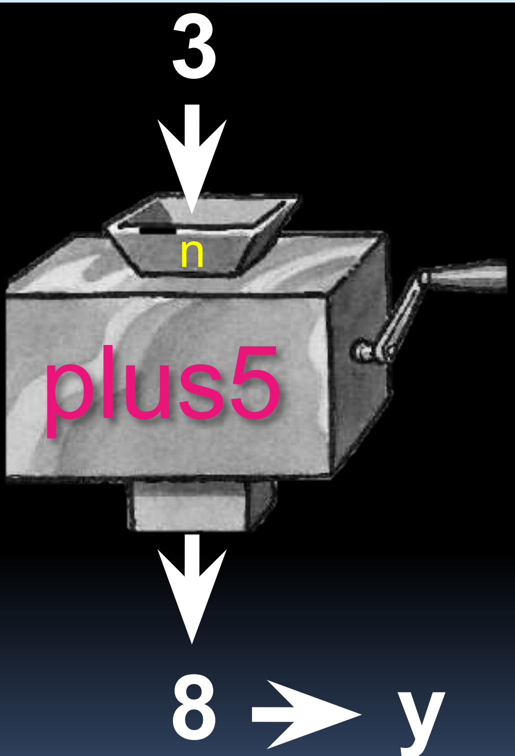
Abstraction: Generalization

REVIEW

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

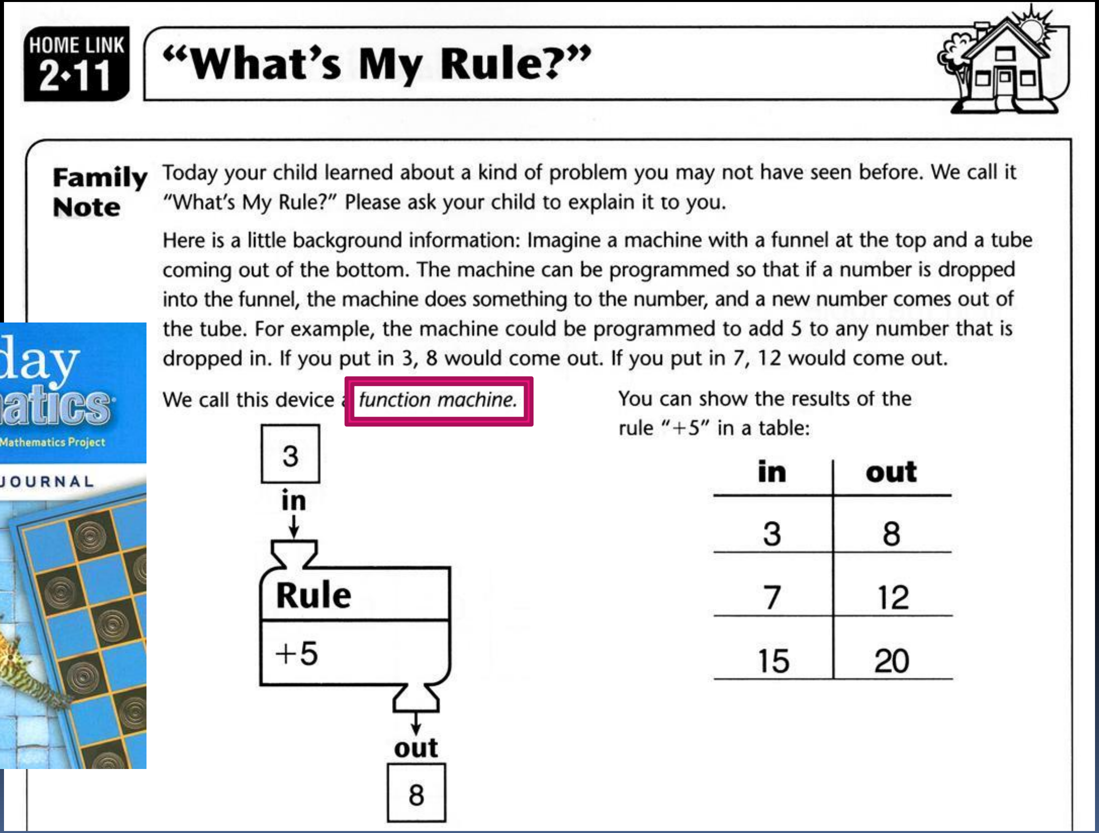
y plus5(n)

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



Function machine
(*Simply Scheme*, Harvey)

Functions in 2nd Grade Math Curricula!



HOME LINK 2•11

"What's My Rule?"

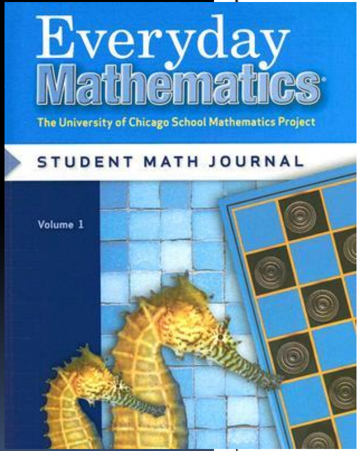
Family Note Today your child learned about a kind of problem you may not have seen before. We call it "What's My Rule?" Please ask your child to explain it to you.

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:

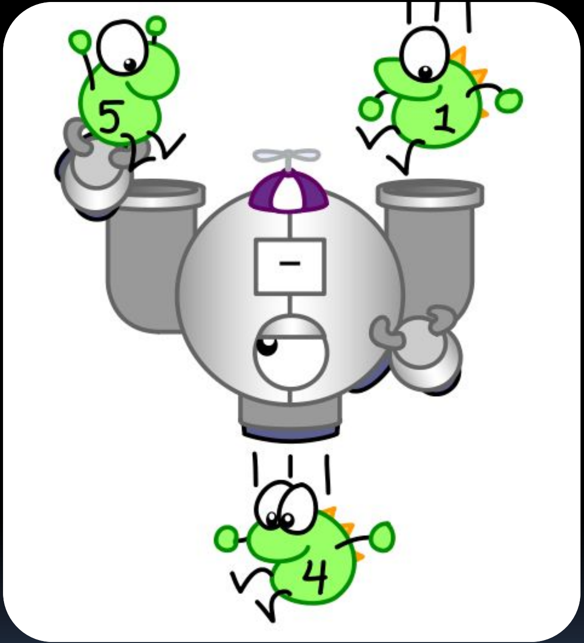
in	out
3	8
7	12
15	20



Garcia

Function Definition

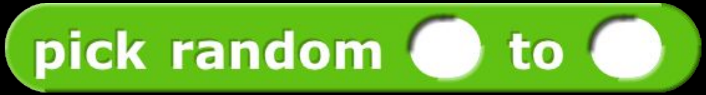
- Functions take in 0 or more inputs, return exactly 1 output
- The same inputs MUST yield same outputs.
  - Output function of input only
- Other rules of functions
  - No state (prior history)
  - No mutation (no variables get modified)
  - No side effects (nothing else happens)

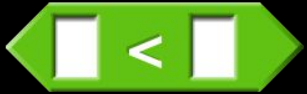


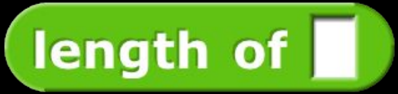
Function Metaphor
(*CS Illustrated*, Ketrina Yim)

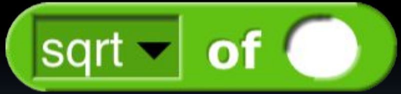
Garcia

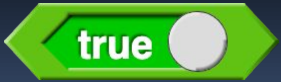
(Cal) Which is NOT a function?

a) 
pick random ( ) to ( )

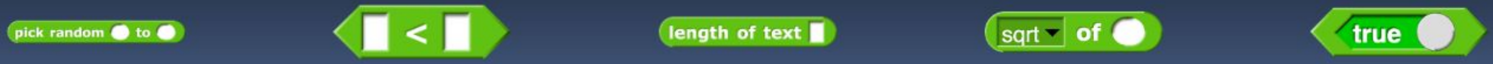
b) 
[ ] < [ ]

c) 
length of [ ]

d) 
sqrt ▾ of ( )

e) 
true

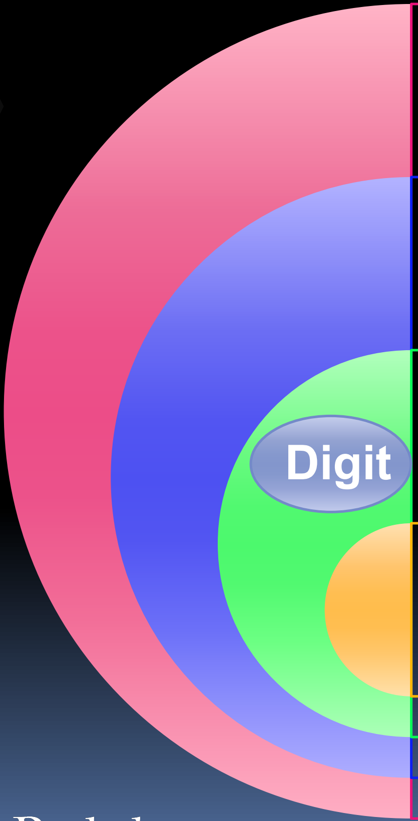
L02a Which is NOT a function?



Data Types Domain & Range

Data Types (you'll make too!)

Boolean
(True or False)



Sentences	• Words separated by N spaces, N ≥ 0 • E.g., CS 10 is great
Word	• Any # of symbols • Length ≥ 1, no spaces • E.g., Cal, 42, CS10
Character	• Any one symbol • Length = 1 • E.g., A, 3, #
Letter	• A-Z,a-z only • E.g., h, G

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
  - _ < _

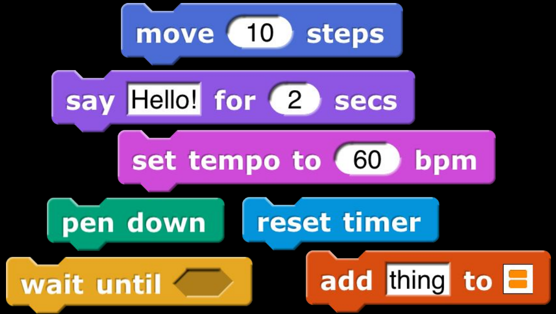
    - Boolean (true or false)
  - _ and _

    - Boolean (true or false)

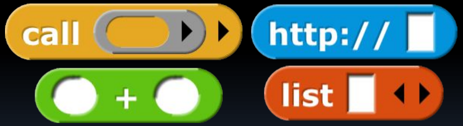
Types of Blocks in Snap!

Procedures, Subroutines

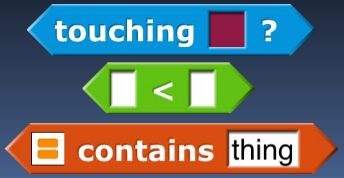
- Command
  - No outputs, meant for side-effects
  - Not a function…
- Reporter (Function)
  - Any type of output
- Predicate (Function)
  - Boolean output
    - (true or false)



```
move (10) steps
say [Hello!] for (2) secs
set tempo to (60) bpm
pen down
reset timer
wait until <>
add [thing] to
```



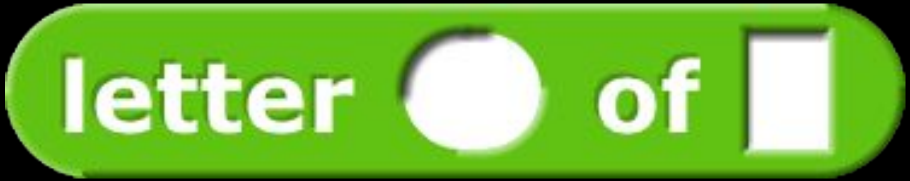
```
call ( ) ▸
http:// [ ]
( ) + ( )
list [ ] ◂▸
```



```
<touching [ ] ?>
<[ ] < [ ]>
< contains [thing]>
```

Garcia

(Cal) Domain, Range of…

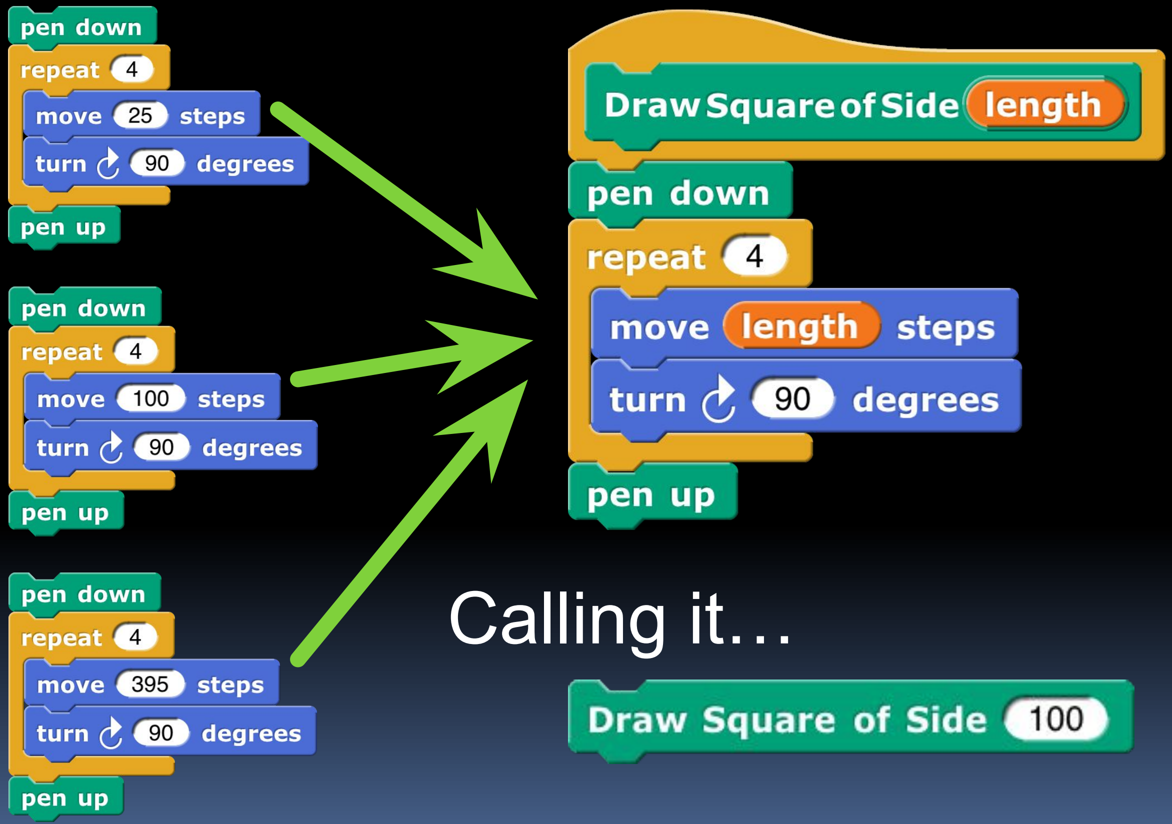


L02b (D)omain, (R)ange of "letter ( ) of [ ]" block

- D: Integer ≥ 1, Number; R: Digit
- D: Integer ≥ 1, Number; R: Letter
- D: Integer ≥ 1, Number; R: Character
- D: Integer ≥ 1, Word; R: Digit
- D: Integer ≥ 1, Word; R: Letter
- D: Integer ≥ 1, Word; R: Character
- D: Integer ≥ 1, Sentence; R: Digit
- D: Integer ≥ 1, Sentence; R: Letter
- D: Integer ≥ 1, Sentence; R: Character

Why Should You Use Functions?

Why Build Blocks? (1/3)



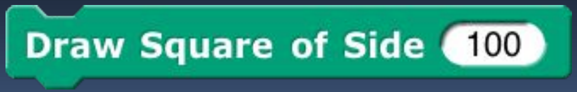
```
pen down
repeat 4
    move 25 steps
    turn ↻ 90 degrees
pen up
```

```
pen down
repeat 4
    move 100 steps
    turn ↻ 90 degrees
pen up
```

```
pen down
repeat 4
    move 395 steps
    turn ↻ 90 degrees
pen up
```

```
Draw Square of Side (length)
pen down
repeat 4
    move (length) steps
    turn ↻ 90 degrees
pen up
```

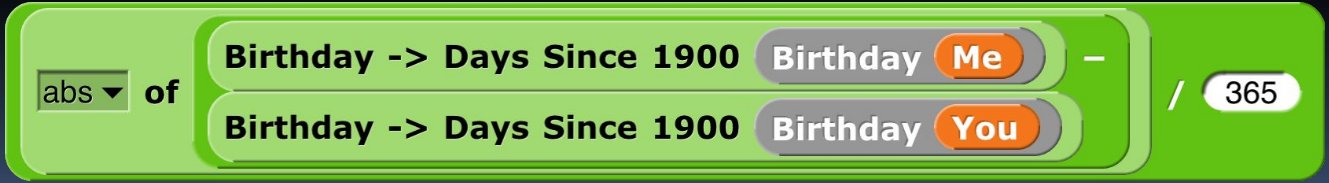
Calling it…



```
Draw Square of Side 100
```

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



```
abs of ((Birthday -> Days Since 1900 (Birthday Me)) - (Birthday -> Days Since 1900 (Birthday You))) / 365
```

en.wikipedia.org/wiki/Functional_programming

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.

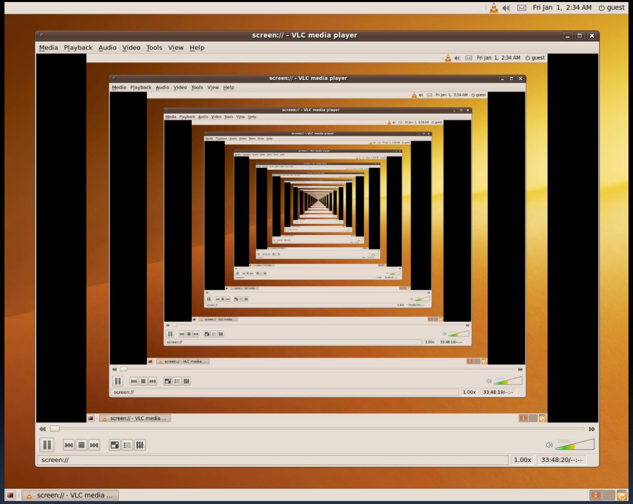
Cabinet Aisle in a Datacenter (*Wikipedia*, Robert Harker)

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)


screen:// - VLC media player
Media Playback Audio Video Tools View Help
Fri Jan 1, 2:34 AM
guest
screen://
1.00x
33:48:20/--:--
screen:// - VLC media ...

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

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