Quest (half quiz half test) in **in 7 days!!**

CS10 : The Beauty and Joy of Computing

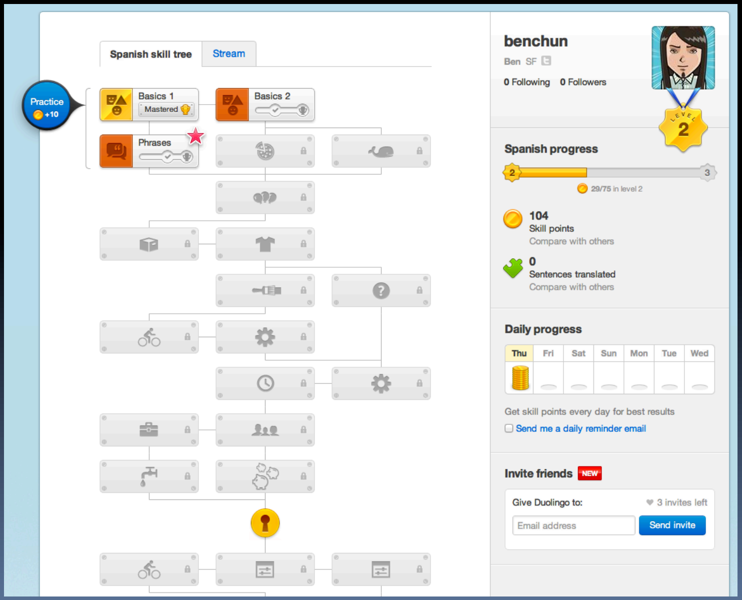
Lecture #4 Functions

UC Berkeley EECS
Summer Instructor
Ben Chun

2012-06-21

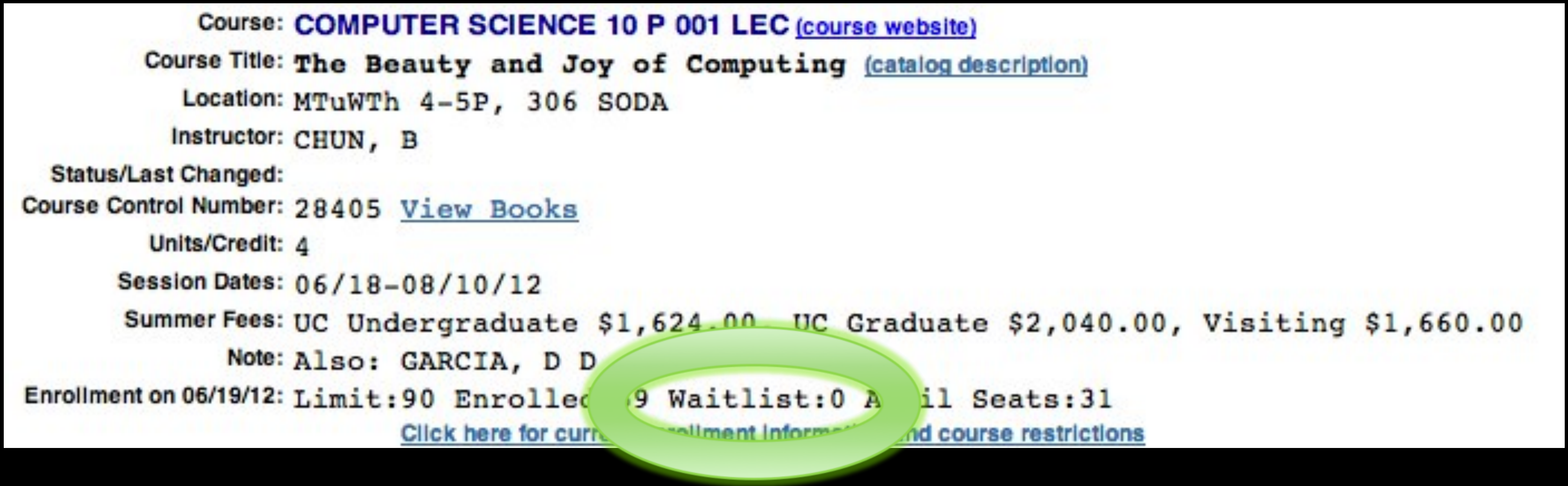
DUOLINGO OPEN TO THE PUBLIC

Luis von Ahn wants to translate the whole web into every major language. Duolingo is doing it using a GWAP! You get points for completing language lessons, and also for translating text from the web. They claim people learn as well as with Rosetta Stone, but Duolingo is free.



http://nyti.ms/Lee58T

Enrollment – everyone IS in



Course:	COMPUTER SCIENCE 10 P 001 LEC (course website)
Course Title:	The Beauty and Joy of Computing (catalog description)
Location:	MTuWTh 4-5P, 306 SODA
Instructor:	CHUN, B
Status/Last Changed:	
Course Control Number:	28405 View Books
Units/Credit:	4
Session Dates:	06/18-08/10/12
Summer Fees:	UC Undergraduate $1,624.00, UC Graduate $2,040.00, Visiting $1,660.00
Note:	Also: GARCIA, D D
Enrollment on 06/19/12:	Limit:90 Enrolle[illegible]9 Waitlist:0 A[illegible]il Seats:31

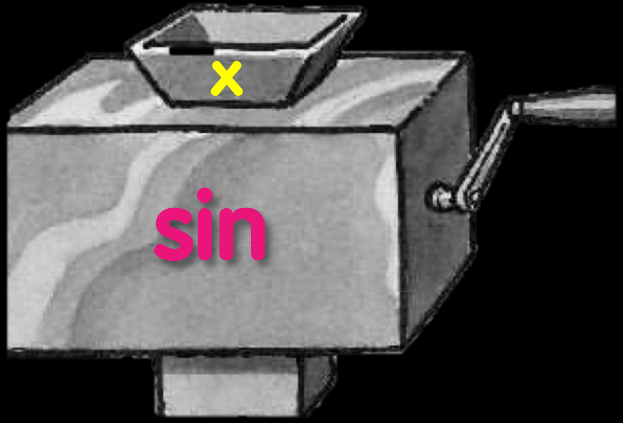
Click here for curr[illegible]rollment inform[illegible]nd course restrictions

Generalization (in CS10) REVIEW

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

  y = sin(x)

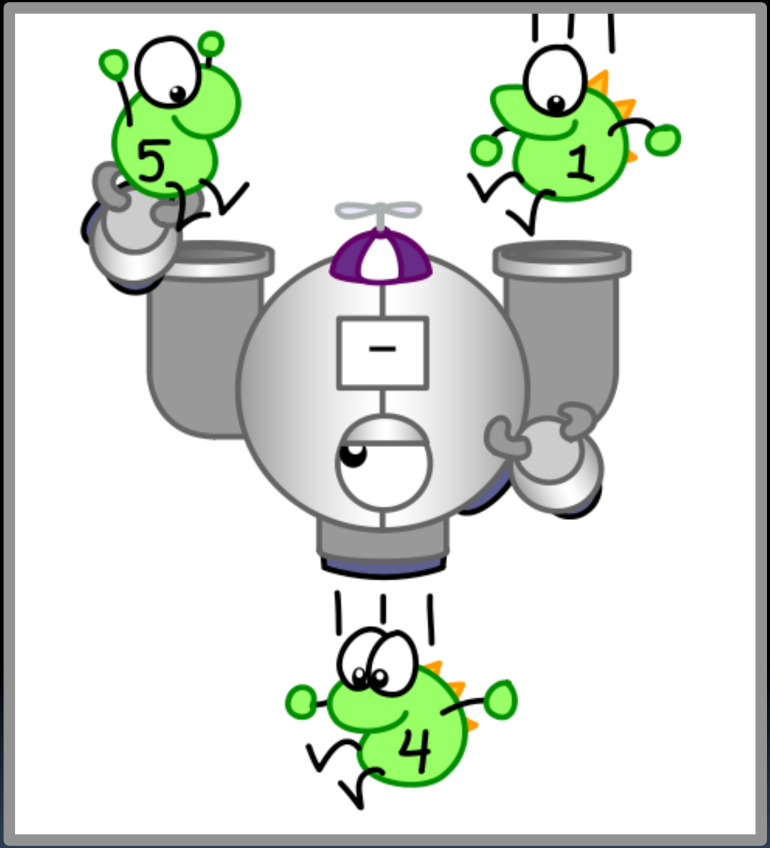
  - sin is the function
  - x is the input
  - It returns a single value, a number



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

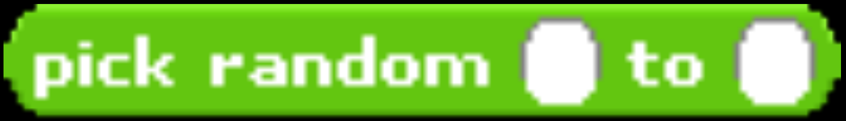
Function basics

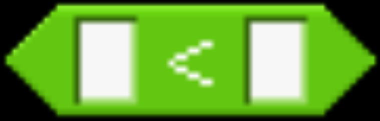
- Functions take in 0 or more inputs and return exactly 1 output
- The same inputs MUST yield same outputs.
  - Output function of input only
- Other rules of functions
  - No state (prior history)
  - No mutation (no variables get modified)
  - No side effects (nothing else happens)

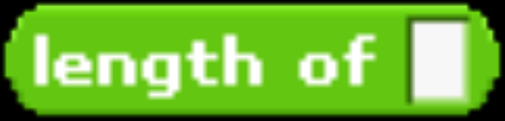


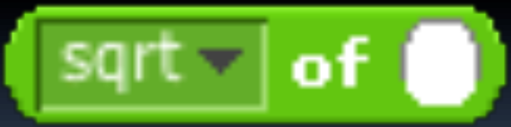
CS Illustrated function metaphor

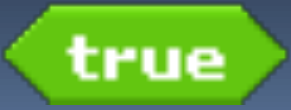
Which is NOT a function?

a) 
pick random ( ) to ( )

b) 
< [ ] < [ ] >

c) 
length of [ ]

d) 
[sqrt ▾] of ( )

e) 
< true >

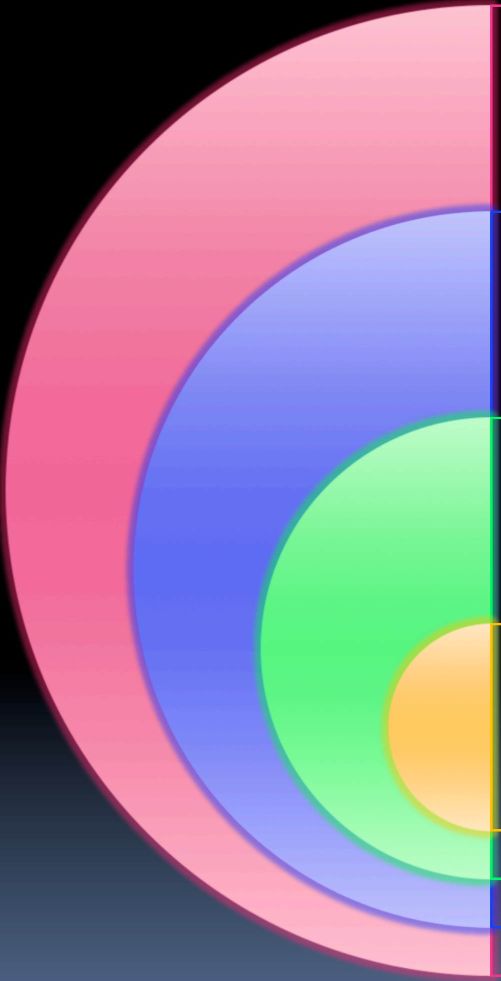
Chun, Summer 2012

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)



Sentences
- Words separated by N spaces, N ≥ 0
- E.g., CS 10 is great

Word
- Length ≥ 1, no spaces
- Cal, 42, CS10

Character
- Length = 1
- E.g., A, 3, #

Digit
- 0-9 only
- E.g., 7

Chun, Summer 2012

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.

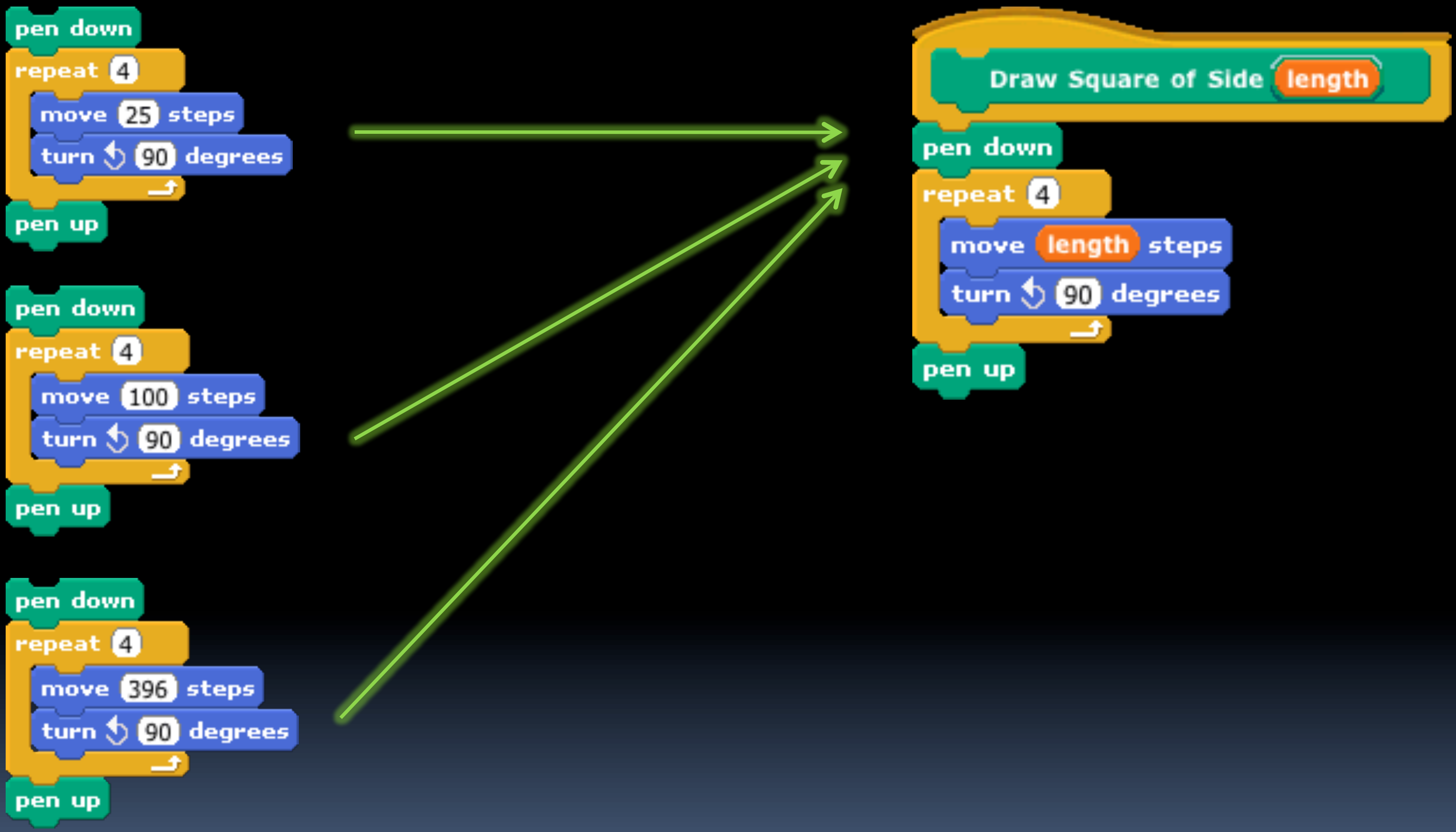
Datacenter

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)



```
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 396 steps
    turn ↻ 90 degrees
pen up
```

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

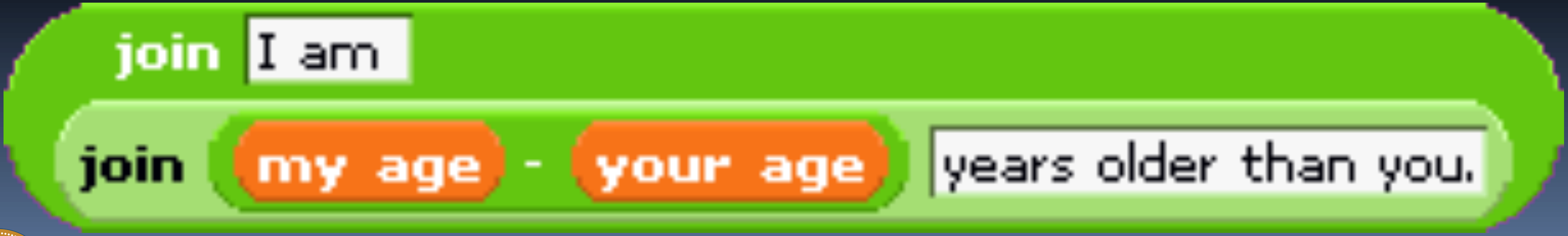
The power of **generalization**!

Why use functions? (2)

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

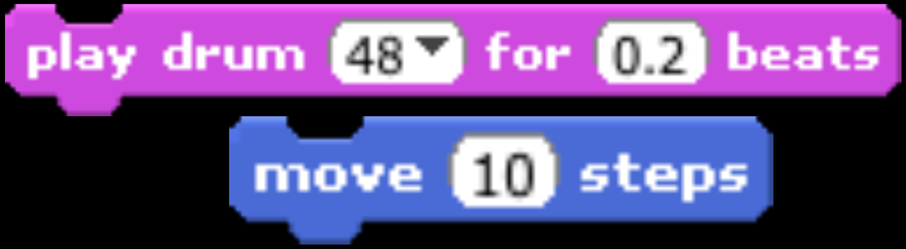
They are literally the **building blocks of almost everything** that we create when we program.

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

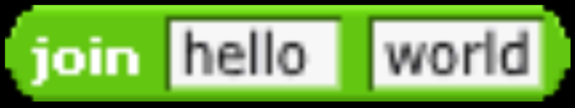


Types of Blocks

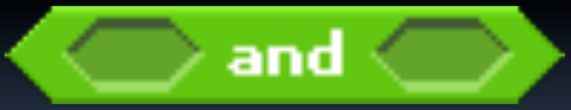
- **Command**
  - No outputs
  - Used for side-effects



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



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

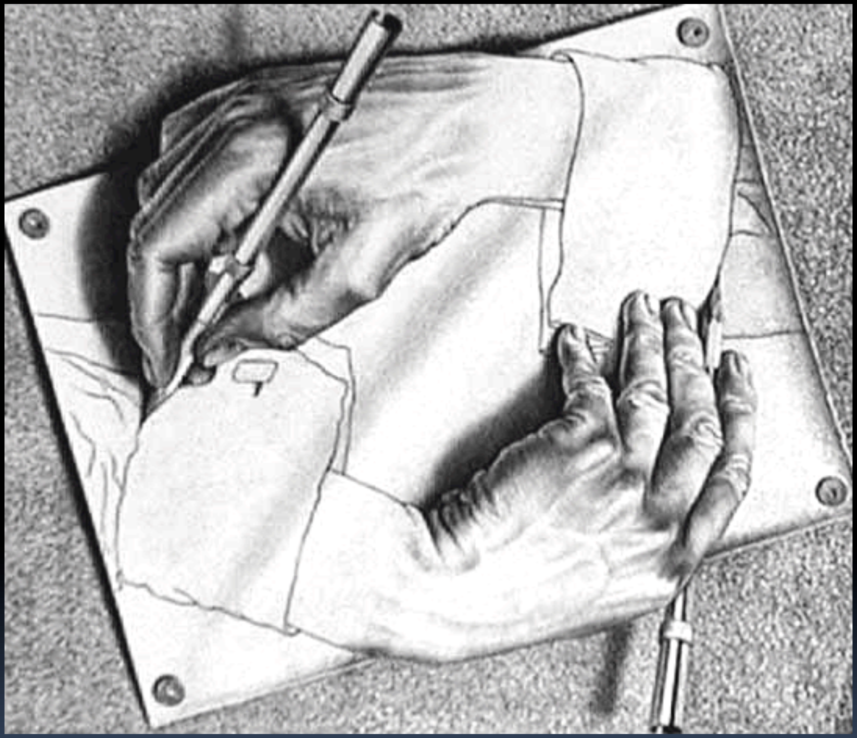


Quick Preview: Recursion

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

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