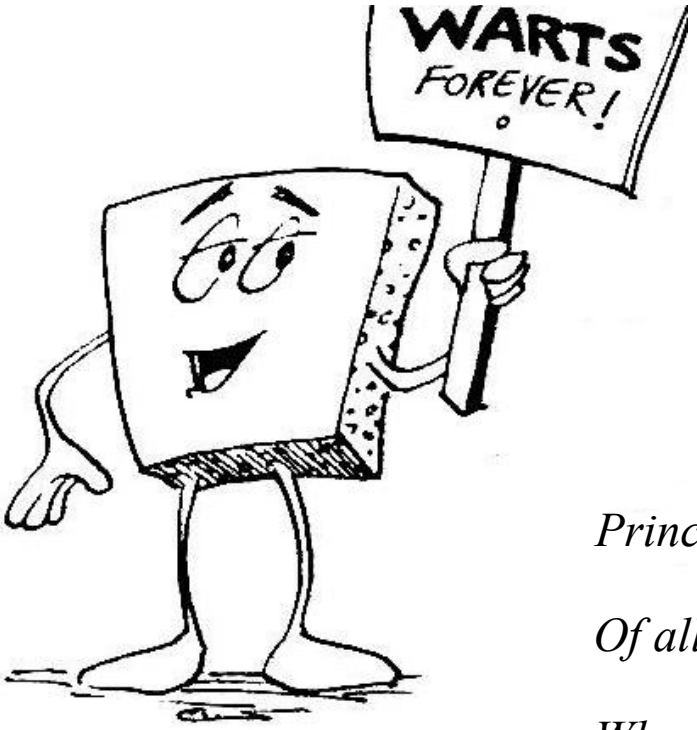
Welcome to CS 60 !



an advocate of
concrete computing

Principles of CS

Principles of CS gets two thumbs.

Of all the classes I took, this was one of them.

When CS 60 was over, I knew it was a good thing.

- Megacritic's course reviews

Some of your hosts…





Introductions...

Zach Dodds
Olin 1255
dodds@cs.hmc.edu

fan of low-level AI
fan of Starbucks

and not afraid of stuffed animals!

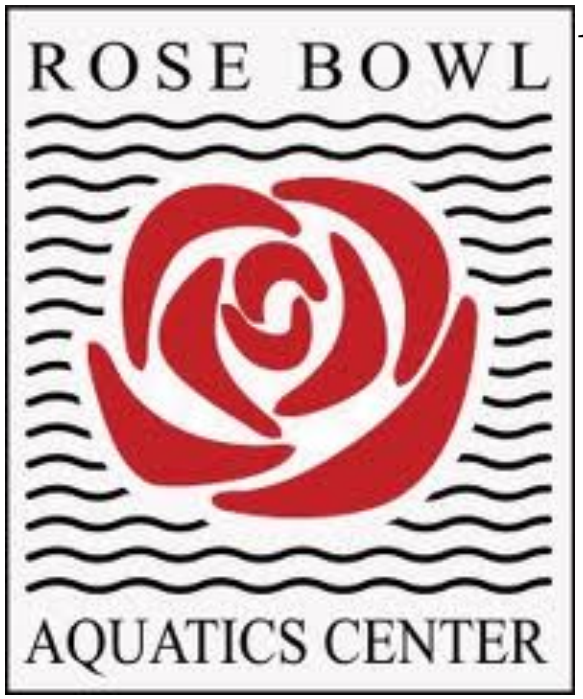
Introductions...

Zach Dodds
Olin 1255
dodds@cs.hmc.edu

fan of low-level AI
fan of Starbucks

And McDonalds – anywhere!

Introductions...



Colleen Lewis
Olin 1241
lewis@cs.hmc.edu

fan of *swimming/sleeping*
fan of *baking*

How I spent my summer vacation…

Finishing my PhD at Berkeley...

...teaching 6th graders computer science!

SCRATCH

CS 60 vs. CS 5

CS 5's goals

Solving computational problems in *one language* (Python)

Conveying some of CS's **breadth**

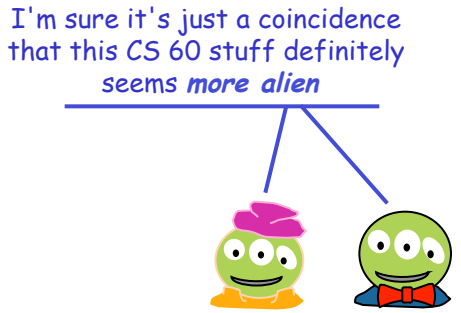
scheduled labs?

CS 60's goals

Solving computational problems in *several different languages!*

Conveying more of CS's **depth**

Understanding computational problems
their difficulty and limitations
solution principles and *efficiency*



Syllabus, briefly

Lectures **MW :** 1:15-2:30 pm

Key skills, topics, and their motivation

Insight into the HW problems (what, ***why***, how)

Required! Let me know if you won't make it

laptops *not* encouraged

Website `http://www.cs.hmc.edu/courses/2012/fall/cs60/`

Tutors and tutoring hours **LOTS!!** `See website...`

Email help `Piazza!` copies to us and to all of the tutors…

Office Hours **Fri.** 2:00-4:00 (Z.D.) **Mon.** (C.L.)

Fridays ~ come by the LAC computer lab

anytime: contact us by email or stop by...

Mondays	Tuesdays	Wednesdays	Thursdays	Fridays	Saturdays	Sundays
in LAC	**in LAC**	**in LAC**	**in LAC**	**in LAC**	**in LAC**	**in LAC**
6-8pm (up to 2):	**8-10pm (up to 2):**	**7-9pm (up to 1):**	**7-9pm (up to 1):**	**1-3pm (up to 1):**	**1-3pm (up to 1):**	**1-3pm (up to 1):**
Helen Woodward	Bridgette Eichelberger	Emma Davis	Mark Mann	Eoin Nugent	Alistair Dobke	Nick Carter
Yukun Lin						
Viona Lam				**3-5pm (up to 1):**	**3-5pm (up to 1):**	**3-5pm (up to 1):**
	10-midn. (up to 2):			Jane Hoffswell	Nabil Zaman	Brett Mills
8-10pm (up to 2):	Tuan Nguyen			Sarah J		
Sidra Hussain						**6-8pm (up to 2):**
Eoin Nugent						Sidra Hussain
L. St. Marie						
10-midn. (up to 3):						**8-10pm (up to 2):**
Alistair D						Helen W
mmann						Olivia Weissblum
Cecily !						
						10-midn. (up to 2):
						Nabil Zaman

Tutoring hours @ LAC

Keys to CS 60 happiness!

Tutors and tutoring hours LOTS!! See above/website...

Email help Piazza! copies to us and to all of the tutors…

Office Hours **Fri.** 2:00-4:00 (Z.D.) **Mon.** (C.L.)

Fridays ~ come by the LAC computer lab
anytime: contact us by email or stop by...

Grading and HW

Grades

~ 60% Assignments
~ 30% Exams
~ 10% Participation/"quizzes"

```
(define (score p)
  (cond
    ((>= p 0.95) "A")
     (>= p 0.90) "A-")
     ;; and so on...
```

This seems like a crazy *scheme* !

Exams

Midterm 1: Wed. 11/7/12

Final: Fri. 12/21/12

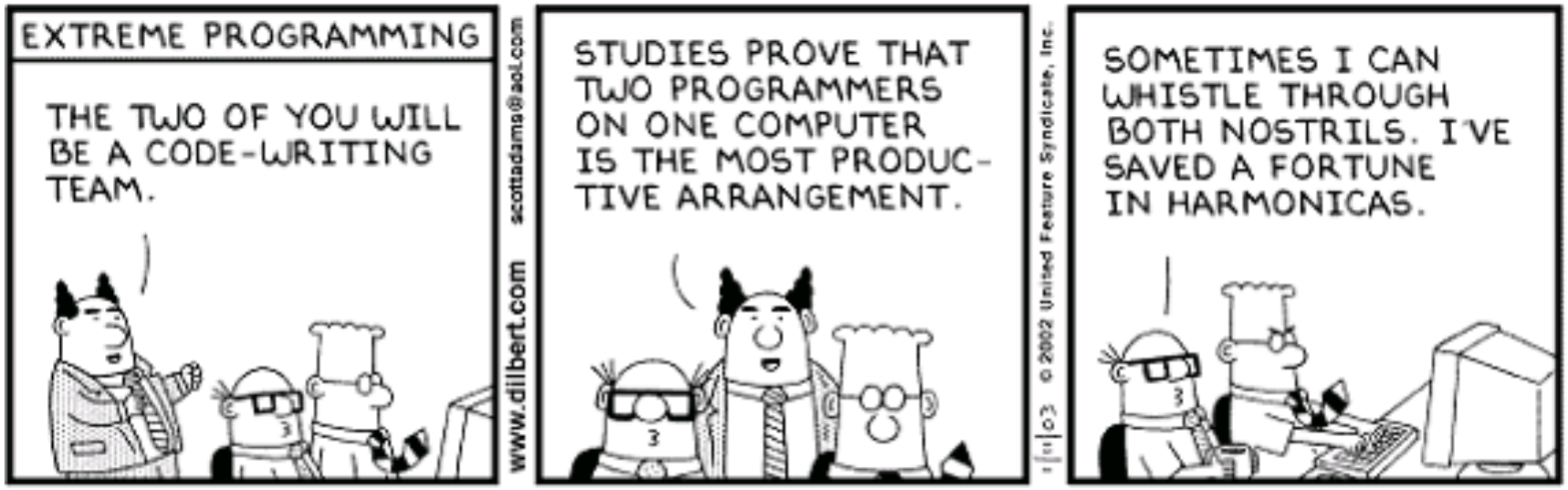
Assignments

"Late Days"

1-10 problems, 100 points per week; You have 3 ***CS 60 Euros*** to use…

Due Monday nights: by 11:59 pm at the submission site: **`asdf`**

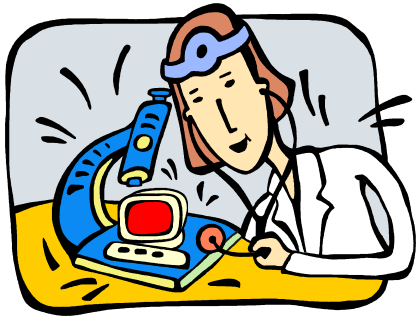
Some problems are ***individual***, some are optionally ***pair*** 



Pairing!

- On some problems, you may pair program – ***do!***
- ***You must practice "pair programming"***
  - both people at the computer at the same time
  - trade off "driver" and "navigator" every 30 mins
  - both people contribute equally and fully to the solution
- Individual problems must be completed individually
- Read the syllabus Collaboration Policy!

Principles of CS?

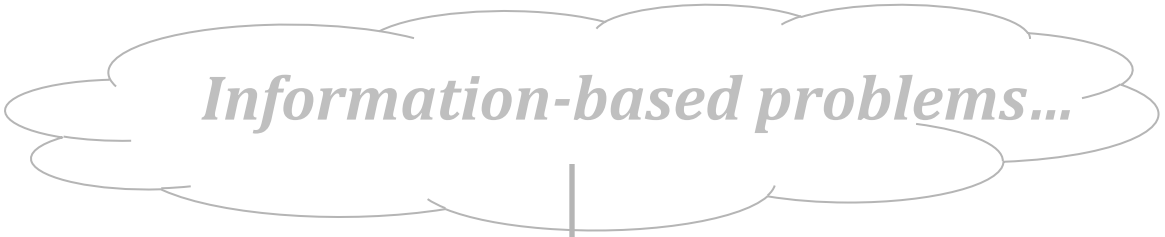


"Computer Science is no more about computers than astronomy is about telescopes.“

— E. W. Dijkstra

	raw material	field
Properties and ***applications***	matter ‘n’ energy	physics
	molecules	chemistry
	cells and organisms	biology
	rule-based systems	mathematics
	information	computer science

Computer Science *Principles*



Algorithms + Programs

Design

- Which language?
- Which data structures?
- What algorithm?
- How to structure my functions?

Analysis

- Does my algorithm solve the problem?
- Is my program bug free?
- How fast does my algorithm run?
- Is the problem solvable at all?

We'll do both of these today & in this week's HW!

Computer Science *Principles*

Subsequence alignment...

AACAGTTACC **TAAGGTCA**

What insertions/deletions would best align these two sequences?

"Left-justify algorithm"

```
AACAGTTACC
TAAGGTCA--
1011001022
```

Total cost: 8

"Gonzo greed algorithm"

```
-AA---CAGTTACC
TAAGGTCA------
20022200222222
```

Total cost: 20

"Try all pairs algorithm"

```
AACAGTTACC
TA-AGGT-CA
1020010201
```

Total cost: 7

Design

- Which language?
- Which data structures?
- What algorithm?
- How to structure my functions?

Analysis

- Does my algorithm solve the problem?
- Is my program bug free?
- How fast does my algorithm run?
- Is the problem solvable at all?

We'll do both of these today & in this week's HW!

JFLAP

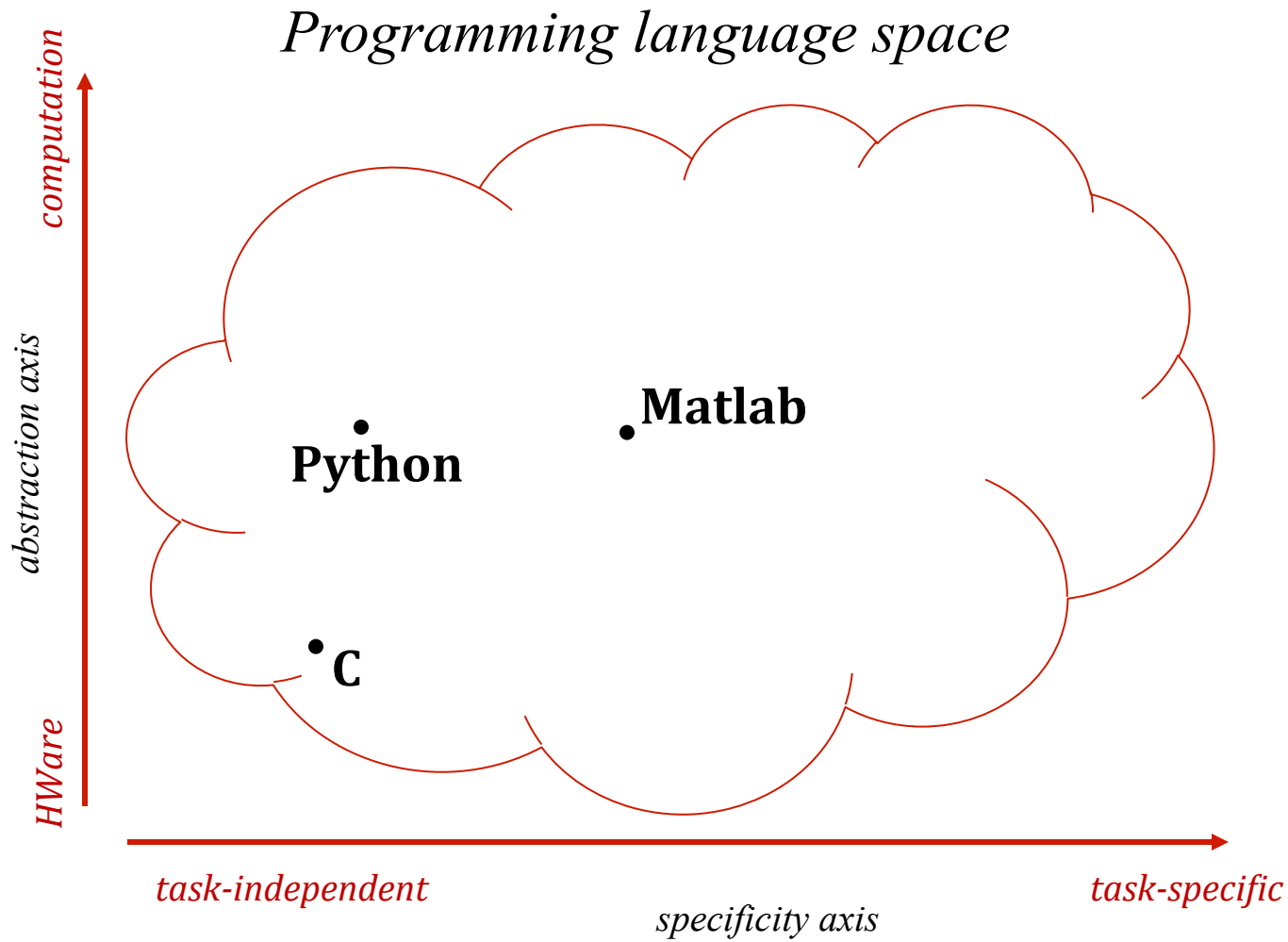
Java

Prolog

Racket

the language formerly known as Scheme

In CS 60 you'll use *at least* four programming languages...



JFLAP

Java

Prolog

Racket

the language formerly known as Scheme

In CS 60 you'll use *at least* four programming languages...

Problem 1: fun!

Quick: An Introduction to Racket with Pictures

This tutorial provides a brief introduction to the Racket programming language by using one of its picture-drawing libraries. Even if you don't intend to use Racket for your artistic endeavours, the picture library supports interesting and enlightening examples. After all, a picture is worth five hundred "hello world"s.

Along the same lines, we assume that you will run the examples using DrRacket. Using DrRacket is the fastest way to get a sense of what the language and system feels like, even if you eventually use Racket with Emacs, vi, or some other editor.

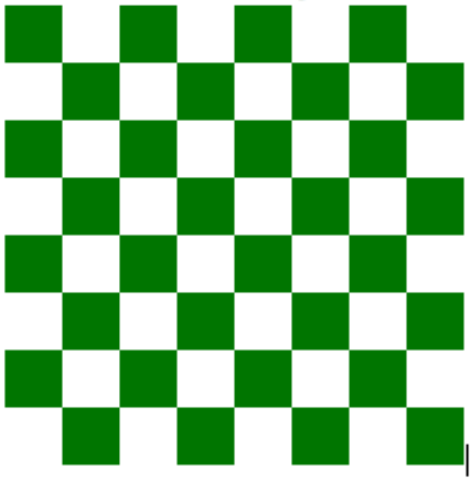
1 Ready...

Download Racket, install, and then start DrRacket.

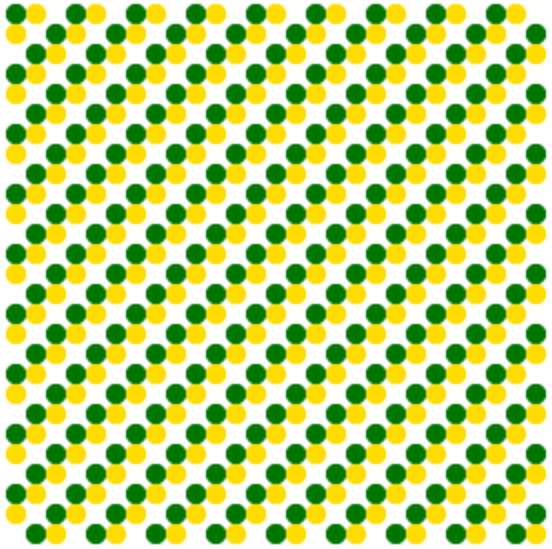
```
> (cboard 10 "gold" "black")
```

```
> (cboard 10 "darkgreen" "white")
```

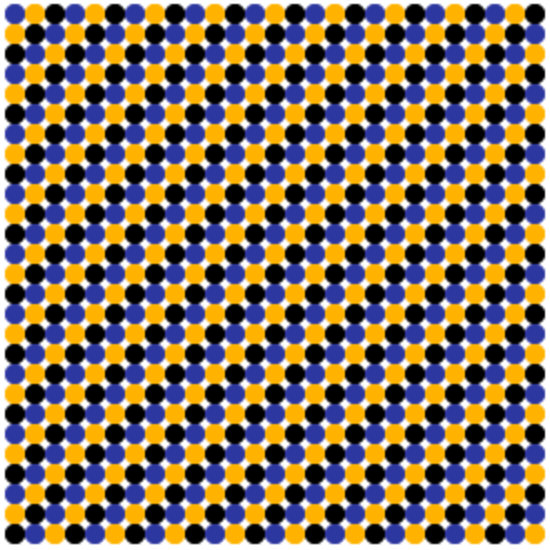
```
> (cboard 25 "darkgreen" "white")
```



```
> (hcomb 10 "darkgreen" "gold" "white")
```



```
> (hcomb 10 "navy" "orange" "black")
```



Problem 2 must be MORE fun!

DEMO: Calling Functions

```
> 3
3
> (+ 3 4)

> (sqrt 16)

> (+ 3 (sqrt 16))

> (+)

> (*)

> (- 1 1 1)
```

Not 3 + 4

Calling two functions? Work from the inside-out

Someone designed these to make sense. EVERYTHING should make sense!

DEMO: Labeling Data

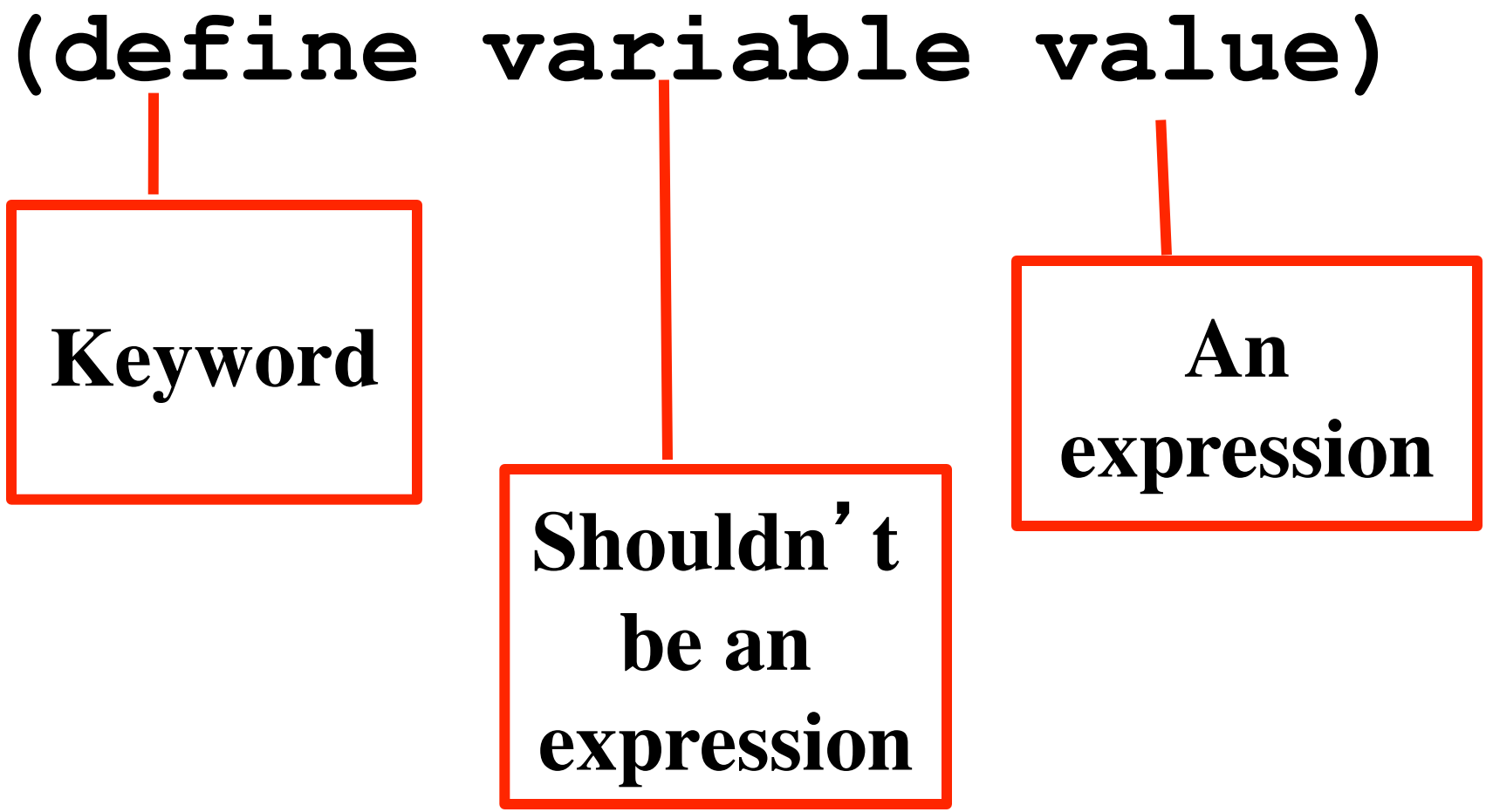
```
> "hello"
"hello"
> hello
hello: undefined;
 cannot reference an identifier before
its definition
> (define hello "how are you?")
> hello
"how are you?"
```

String

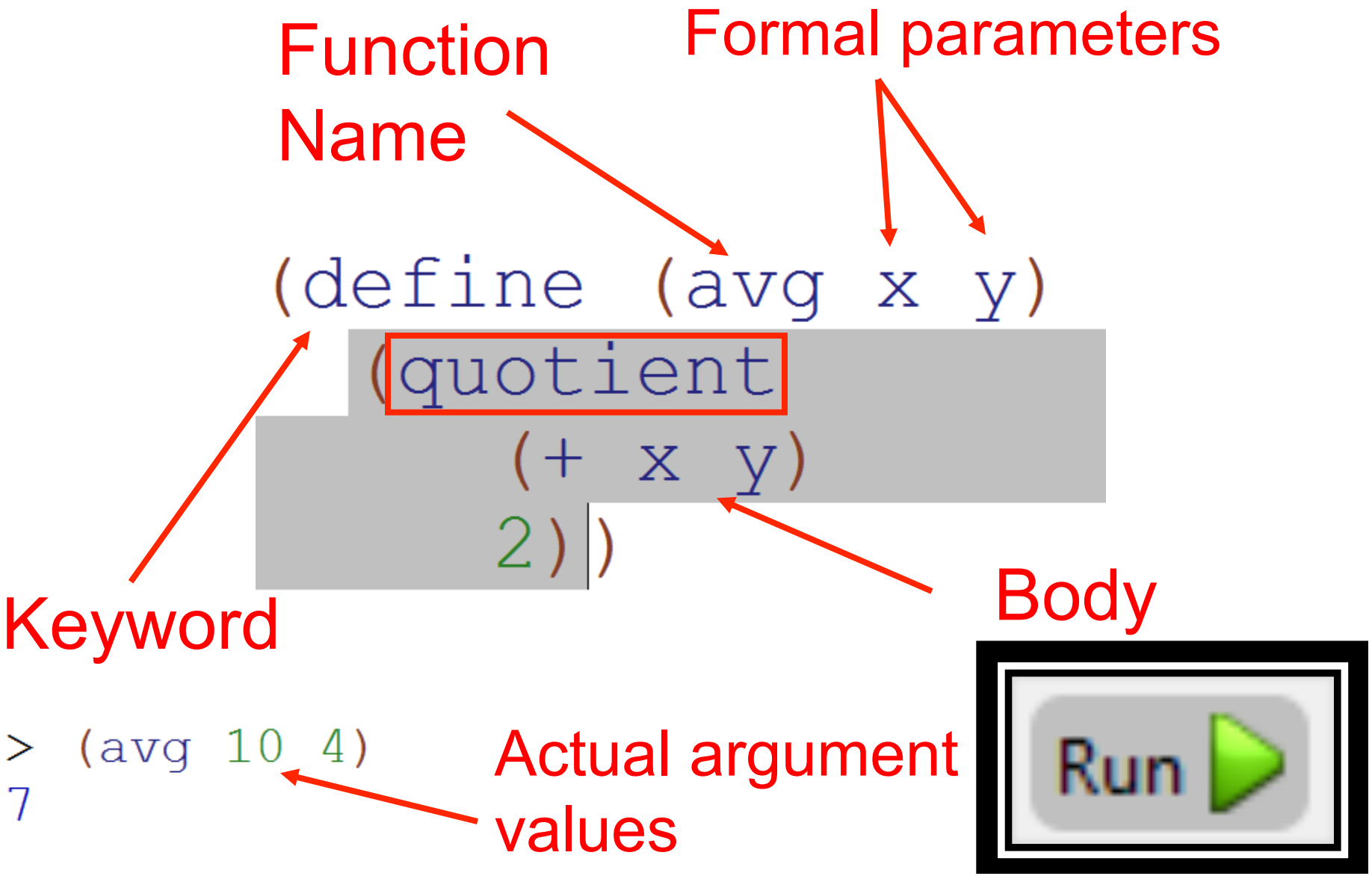
Helpful error messages READ THEM

Label data

How to Label Data



How to label functions



How to use `if` & `cond`

```
(if <predicate>
    <true case>
    <false case> )

(cond
    [ <test1>     <result> ]
    [ <test2>     <result> ]
    [ else        <result> ] )
```

How to use `Let`

(create new variables in definitions)

```
(let
  ( [<variable1> <value1>]
    [<variable2> <value2>]
  )
  <body>
  )
```

DEMO: procedures/parens

```
> (= 2 2)
#t
> (odd? 4)
#f
> (if (odd? 3) "duh" "what?")
"duh"
> +
#<procedure:+>
> avg
#<procedure:avg>
> (7)
application: not a procedure;
 expected a procedure that can be
applied to arguments
  given: 7
  arguments...: [none]
```

`#t` and `#f` are Booleans

Procedures are just things

No extra parens!

Racket thinks the thing after a paren is a function

The `Factorial` function

```
;; fac: the factorial function
;;    inputs: a positive integer, N
;;    outputs: N!
(define (fac N)
  (if (< N 1)
      1
      (* N (fac (- N 1)))
      ))
```

$$\mathbf{1! = 1}$$

$$\boldsymbol{N! = N * (N - 1)!}$$

```
(define (fac N)
  (if (< N 1)
      1
      (* N (fac (- N 1)))
      ))
```

```
(define (fac N)
  (if (< N 1)
      1
      (* N (fac (- N 1)))
      ))
```

```
(define (fac N)
  (if (< N 1)
      1
      (* N (fac (- N 1)))
      ))
```

```
;; add42: adds 42
;;   inputs: an integer, N
;;   outputs: the integer one larger than N
(define (add42 N)
```

```
;; is42: is it Douglas Adams's answer?
;;   inputs: an integer, N
;;   outputs: true if N==42, false otherwise
(define (is42 N)
```

```
;; sign: returns -1, 0, or 1
;;   inputs: an integer, N
;;   outputs: -1 if N<0; 1 if N>0; 0 otherwise
(define (sign N)
  (cond
```

```
;; halve-count:
;;   inputs: an integer, N
;;   outputs: (log base 2 of N), i.e.,
;;            # of times you can divide N by 2
;;            until you reach 1
(define (halve-count N)
```

Try these!

A new programming language might not extend the set of all possible algorithms,

but it *does* extend the set of all algorithms we can *efficiently think about, write, & analyze*...

big-O

Big-O analysis

Problem Find N!

Problem Size value of input, N

Algorithm 5! is 5*4*3*2*1

Racket Code

```
(define (fac N)
   (if (< N 2)
       1
       (* N (fac (- N 1))) ))
```

How many steps are needed *if a "step" is…*

a multiplication | an arithmetic operation | a comparison | a function call

Big - O

…summarizes the asymptotic "Order" of the work done

ignore constant coefficients

ignore all but the biggest term

$N - 1$ is $O(N)$

$2N^2 + 5N$ is $O(N^2)$

$2N^2 + 4N^3$ is $O(\quad)$

Big-O is **the key** to comparing algorithms and problems…

Big-O analysis

Problem	Compute `(halve-count N)`
Problem Size	value of input, N
Algorithm	If N = 11, then 11-5-2-1 takes **3** steps
Racket Code	

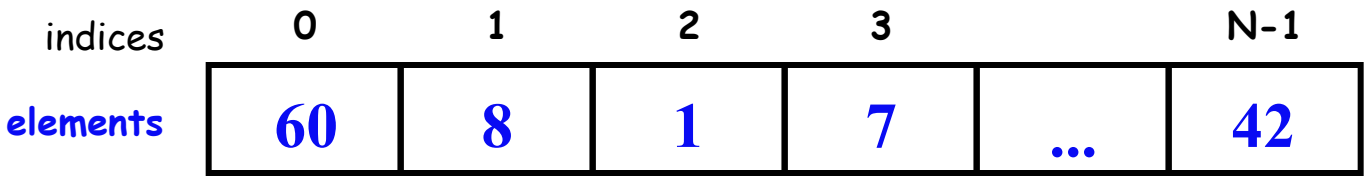
```
(define (h-c N)
   (if (equals? 1 N)
       0
       (+ 1 (h-c (quotient N 2)))
```

How many steps are needed *in big-O terms ?*

does it matter what a "step" is here?

Big-O analysis

Problem	Find the ***min*** of an N-element list
Problem Size	length of list, N
Algorithm	Walk the list!?

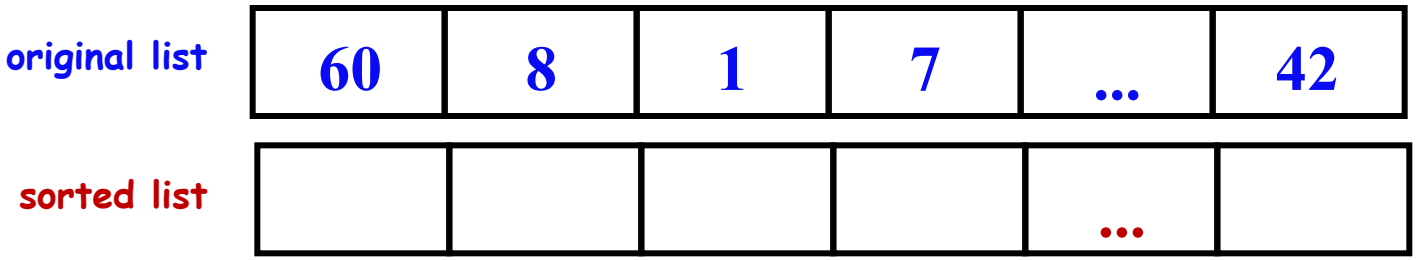


How many steps are needed *in big-O terms ?*

a "step" is often one comparison

Big-O analysis

Problem	***Sort*** an N-element list
Problem Size	length of list, N
Algorithm	Keep finding the next ***min***!



How many steps are needed *in big-O terms ?*

a "step" is often one comparison

"Quiz"

Name:

Birthday:

Your favorite _____ is _____ .

Your *least* favorite _____ is _____ .

Estimate the worst-case big-O complexity for these problems:

1 **Prob:** Find the minimum of a list
Size: N = # elements in the list
Count: the number of comparisons...

O(N)

2 **Prob:** Sort a list via repeated min-finding
Size: N = # elements in the list
Count: the number of comparisons...

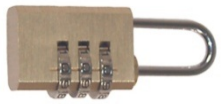
O(N^2)

3 **Prob:** Compute some base to a power: b^N
Size: N = the integer exponent
Count: multiplications...

4 **Prob:** Find the dot product of two vectors
Size: The vectors are both of length N.
Count: multiplications and additions …

5 **Prob:** Guess (and find) a hidden integer X, from 1 to N
- For each guess G, you're told G>X or G<X or G=X.
Count: the number of guesses required…

6 **Prob:** Open a combination lock like these below:
Size: N = # of digits in the combination
Count: the number of guesses required…



N = 3

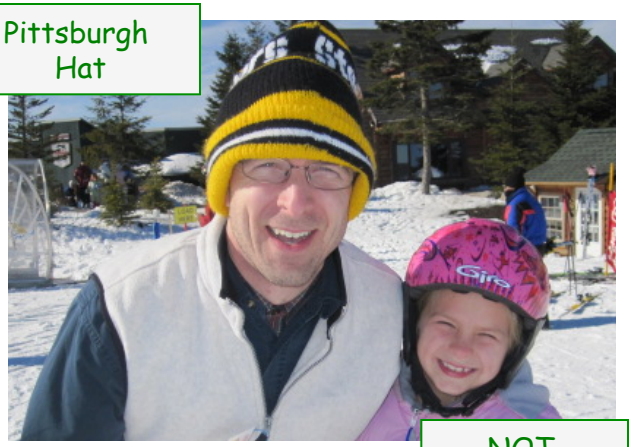
N = 4

7 **Prob:** Multiply two square matrices: ( ) ( ) = ( )
Size: N = one dimension = one matrix side
Count: multiplications and additions...

8 **Prob:** Find the median of a list
Size: N = length of the list
Count: the number of comparisons

9 **Prob:** Determine if a program has an infinite loop.
Size: N = # of characters in the program
Count: all operations.

My "Quiz"



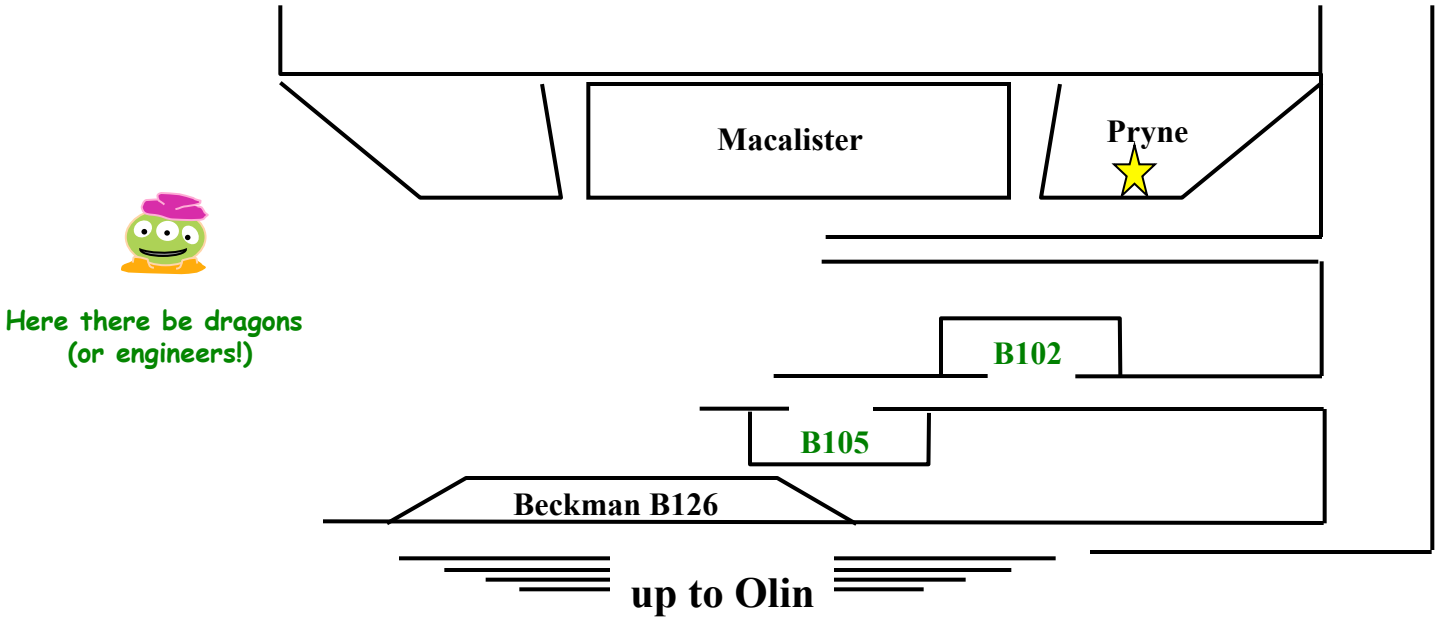
- Name Zachary Dodds
- Birthday 1/21/1969
- A place you considered home Pittsburgh, PA
- Your favorite tv drama _____ is White Collar _____.
- Your least favorite coffee _____ is decaffeinated _____.

What is something you have in common that you didn't know before?

Our taste in hats!

Assignment #0 Due 9/10 11:59pm

- Implementing, testing, and analyzing several *Racket* functions
- Using your machine OR the CS labs: B102, B105 or CIS labs
- Friday 2-4 and lots of grutor hours in the LAC lab



CS labs' code:

Thought for the day:

Racket is an easy language to learn once you get over your fear of parentheses

... nothing to be afraid of in hw#0 !