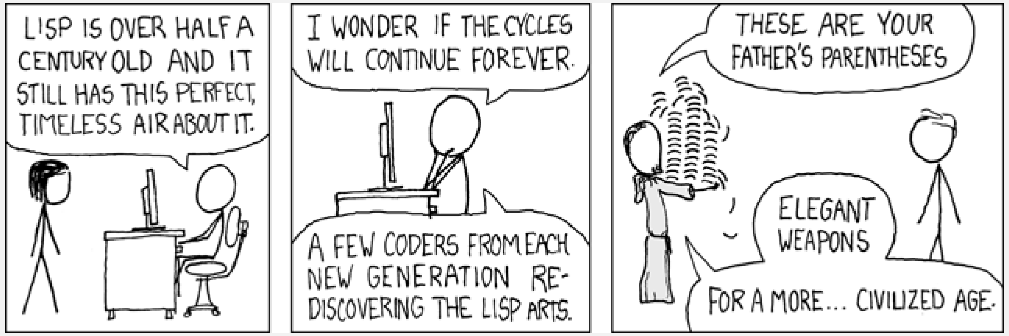
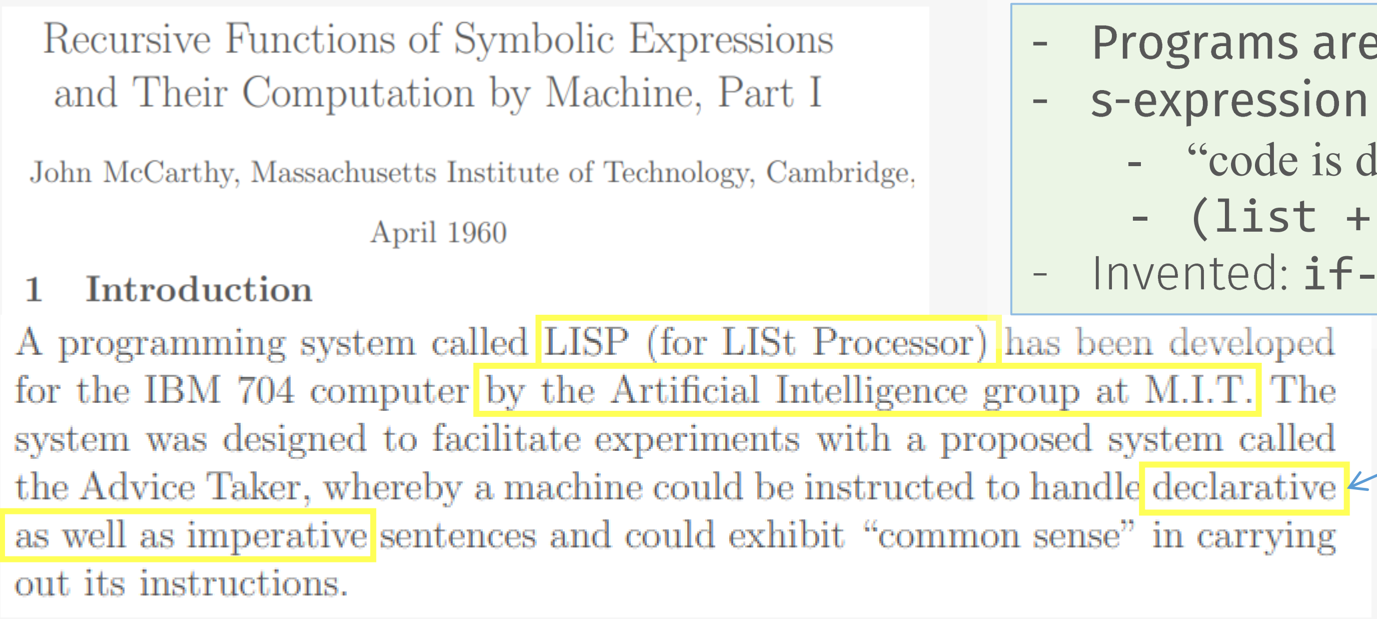
CS450$_{\text{(section 2)}}$

High Level Languages

UMass Boston Computer Science

Wednesday, September 13, 2023


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.



Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

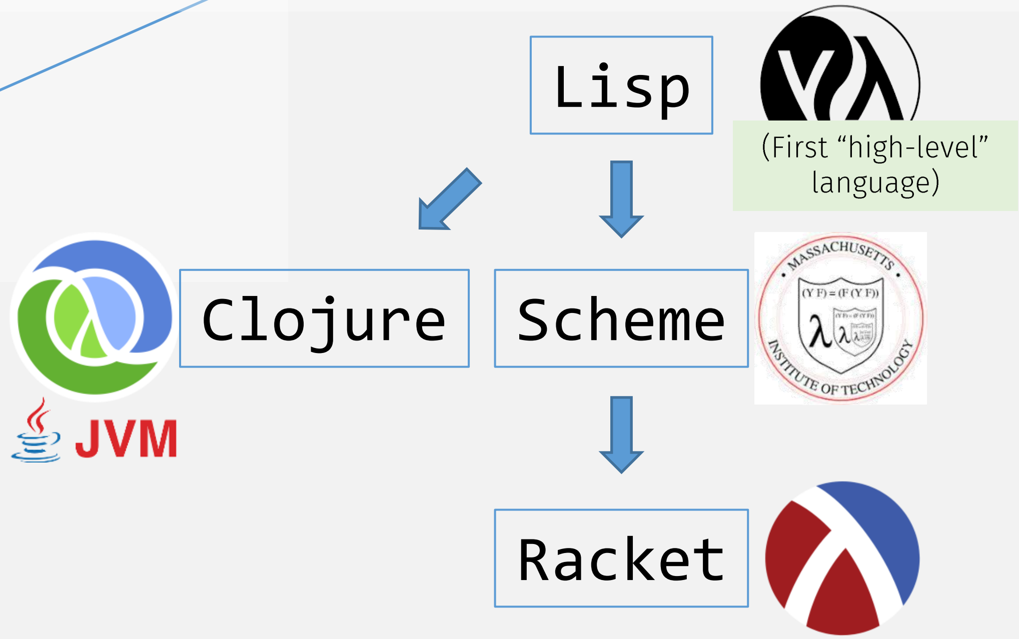
John McCarthy, Massachusetts Institute of Technology, Cambridge,

April 1960

1 Introduction

A programming system called LISP (for LISt Processor) has been developed for the IBM 704 computer by the Artificial Intelligence group at M.I.T. The system was designed to facilitate experiments with a proposed system called the Advice Taker, whereby a machine could be instructed to handle declarative as well as imperative sentences and could exhibit "common sense" in carrying out its instructions.

- Programs are **expressions** (not sequences of instructions!)
- s-expression syntax
  - "code is data, data is code"
  - `(list + 1 2)` is both **program** and **list of chars**
- Invented: **`if-then-else`, `lambda`, recursion, gc** (no ptrs), **`eval`**



Lisp

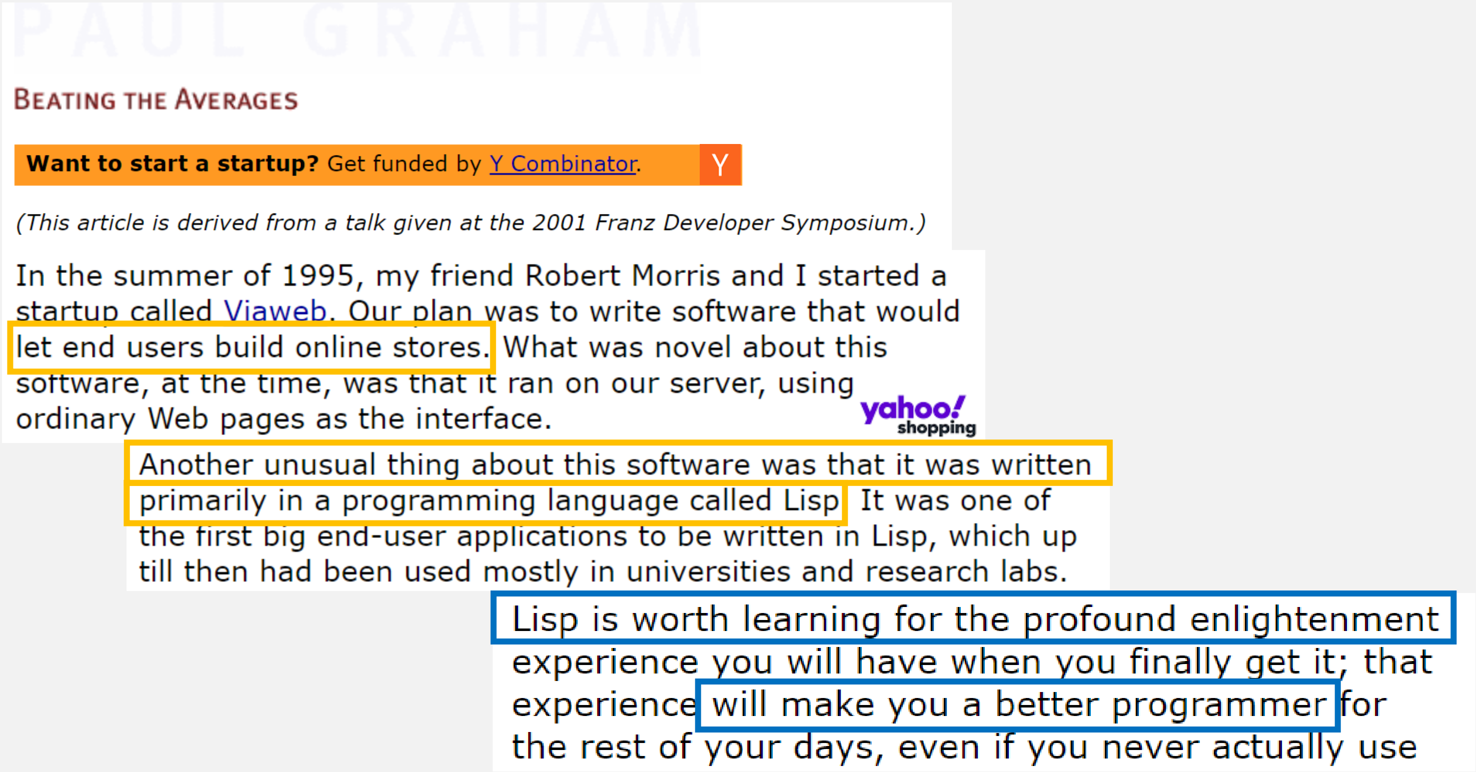
(First "high-level" language)

Clojure

JVM

Scheme

Racket



PAUL GRAHAM

Beating the Averages

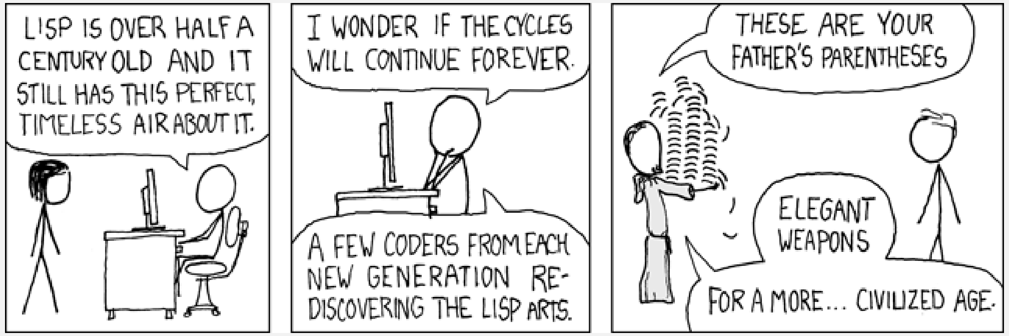
(This article is derived from a talk given at the 2001 Franz Developer Symposium.)

In the summer of 1995, my friend Robert Morris and I started a startup called Viaweb. Our plan was to write software that would let end users build online stores. What was novel about this software, at the time, was that it ran on our server, using ordinary Web pages as the interface.

yahoo! shopping

Another unusual thing about this software was that it was written primarily in a programming language called Lisp. It was one of the first big end-user applications to be written in Lisp, which up till then had been used mostly in universities and research labs.

Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

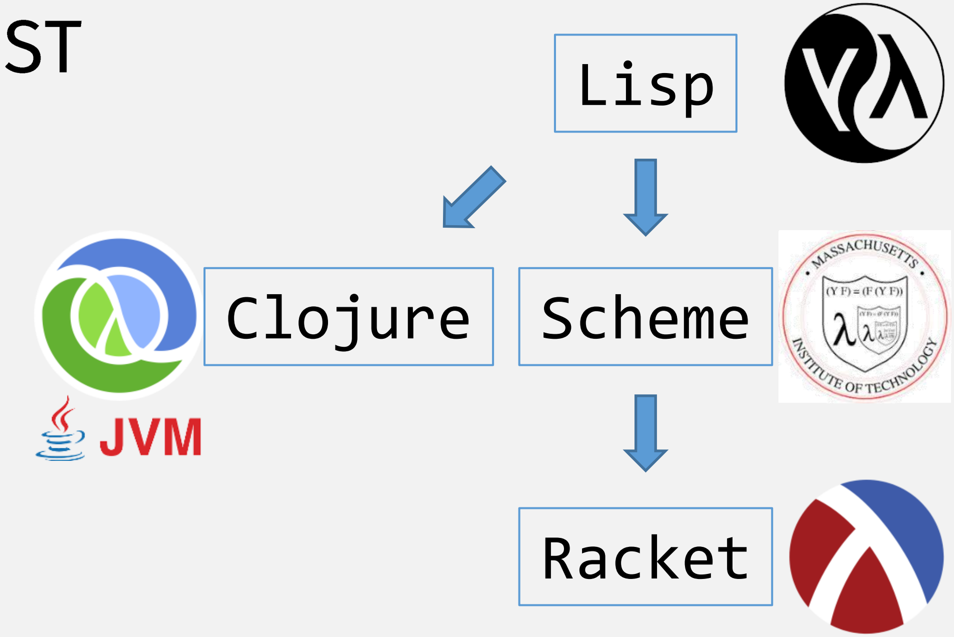
Logistics

- HW 0, part 2 due: **Sun 9/17 11:59 pm EST**

- Course web site:

  `https://www.cs.umb.edu/~stchang/cs450/f23`

  - See new `Racket->Style` section



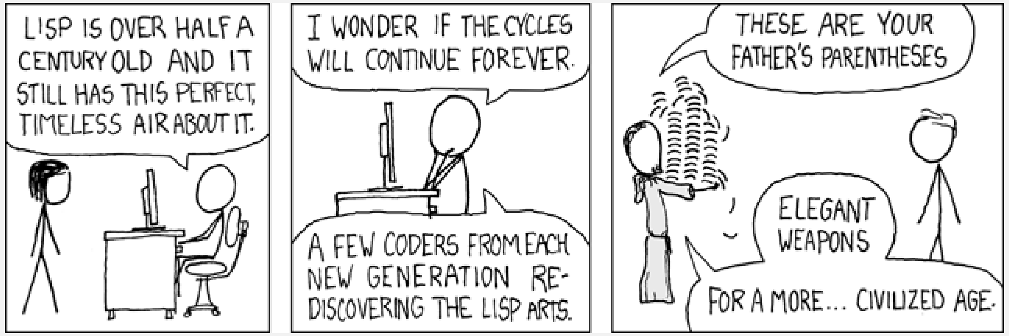
Lisp

Clojure

Scheme

Racket

JVM


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

Functions – code demo

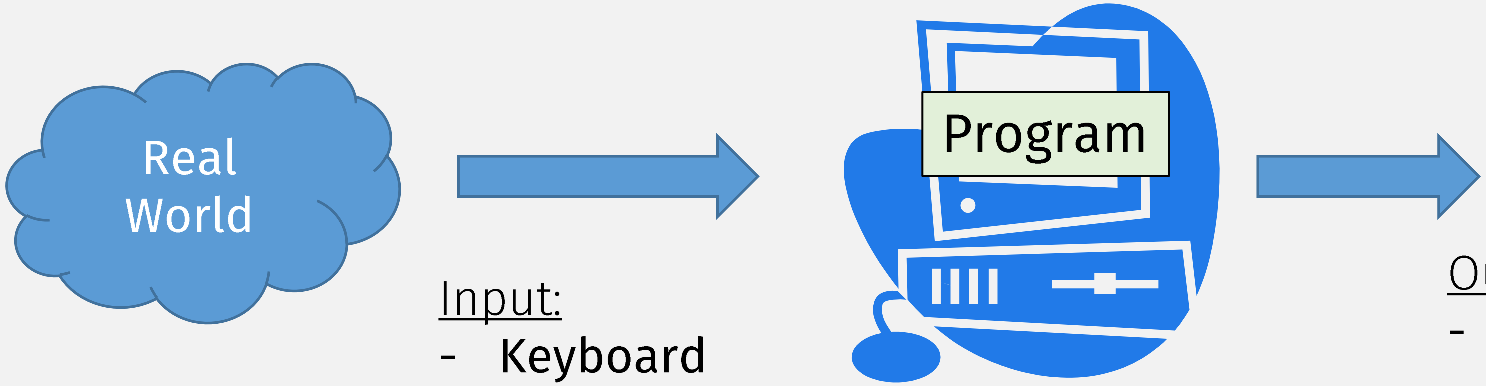
- `define`
  - The only non-expression you should use
- `lambda`
  - (anonymous) function expression
  - Function position in function call is computed expression
  - `((lambda (f) (f f)) (lambda (f) (f f)))`
- Predicates?
  - Function that evaluates to true or false

Programs

- Programs are sequence of `defines` and expressions
  - One of them could be a “`main`” entry point

- When the program is run, each is **evaluated** to get an “answer”
  - similar to “reduction” in math

Programs: Still need I/O



Input:

- Keyboard
- Mouse
- Gamepad
- Touchscreen
- Voice
- File

Output:

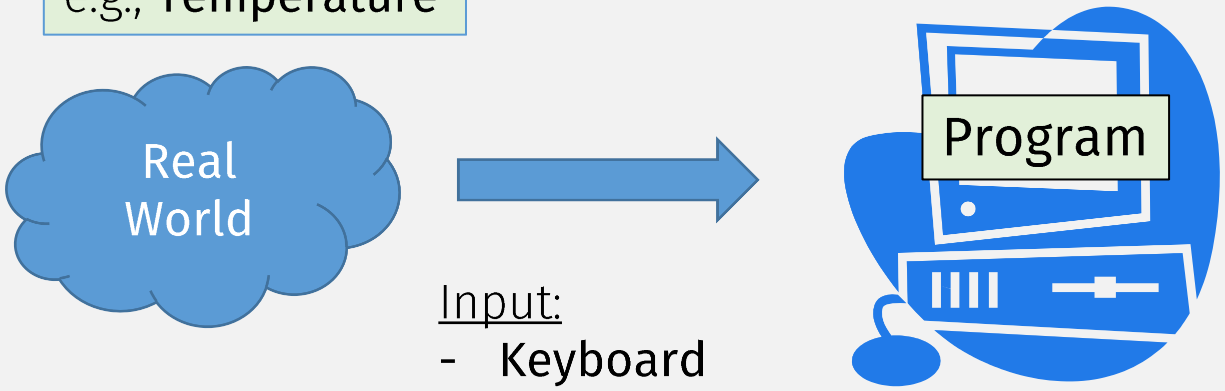
- "answer", e.g., 42
- Picture
- File
- Sound

Program vs Real World

Real World “things” ...

e.g., Temperature

... need a **data representation** in the program



Input:

- Keyboard
- Mouse
- Gamepad
- Touchscreen

When programming, choosing these **data representations** must be the first task! (way before writing any code!!!)

A **Data definition** name

Specify possible values of the data

```
;; A TempC is an Integer
;; Interpretation: It represents a
temperature in degrees Celsius
```

“Interpretation” = Its connection to a real world concept

```
;; A TempF is an Integer
;; Interp: It represents a temperature
in degrees Fahrenheit
```

```
;; A TempK is an non-negative Integer
;; Interp: It represents a temperature
in degrees Kelvin
```

Design Recipe

1. Data Design
    - Define the needed Data Definitions
        - **Data Definitions** are a representation of real world concepts can be manipulated by the program
        - The **Interpretation** explains the connection to the real world

Design Recipe

1. Data Design
2. Function Design(s)

Designing Functions

```
;; A TempC is an Integer
;; Interp: represents a temp in degrees Celsius
;; A TempF is an Integer
;; Interp: represents a temp in degrees Fahrenheit
```

1. Name
2. Signature
   - # of arguments and their data type
   - Output type
   - Use or create new Data Definitions (if needed)

```
;; c2f: TempC -> TempF
;; Converts a Celsius temperature to Fahrenheit
```

3. Description
4. Examples
   - Show how the function works

```
(check-equal? (c2f 0) 32)
(check-equal? (c2f 100) 212)
(check-equal? (c2f -40) -40)
```

5. Code
6. Tests

```
(define (c2f ctemp)
    (+ (/ (* ctemp 9) 5) 32))
```

```
(check-equal? (c2f 1) (+ (/ 9 5) 32))
```

Designing Functions

```
;; A TempC is an Integer
;; Interp: represents a temp in degrees Celsius
;; A TempF is an Intege Rational
;; Interp: represents a temp in degrees Fahrenheit
```

1. Name
2. Signature
   - # of arguments and their data type
   - Output type
   - Use or create new Data Definitions (if needed)

```
;; c2f: TempC -> TempF
;; Converts a Celsius temperature to Fahrenheit
```

3. Description
4. Examples
   - Show how the function works

```
(check-equal? (c2f 0) 32)
(check-equal? (c2f 100) 212)
(check-equal? (c2f -40) -40)
```

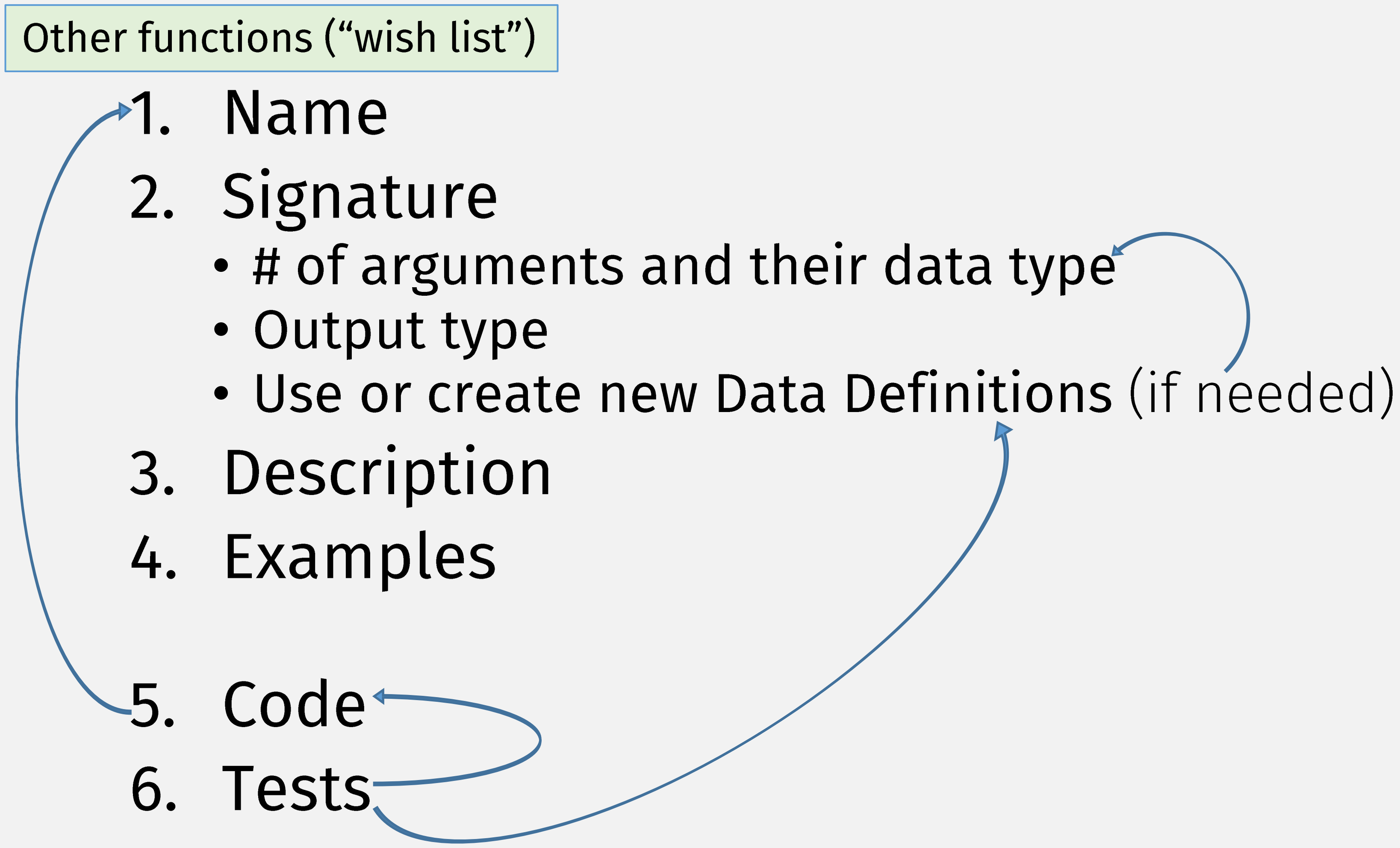
5. Code
6. Tests

```
(define (c2f ctemp)
    (+ (/ (* ctemp 9) 5) 32))
```

```
(check-equal? (c2f 1) (+ (/ 9 5) 32))
```

Something is wrong!
- Code?
- Signature?
- Data Definition?

Iterative Programming



Other functions (“wish list”)

1. Name
2. Signature
   - # of arguments and their data type
   - Output type
   - Use or create new Data Definitions (if needed)
3. Description
4. Examples
5. Code
6. Tests

Programming is an **iterative** process!

```
(require 2htdp/universe)
```

Interactive Programs (with **big-bang**)

- **big-bang** starts an (MVC-like) interactive loop
  - repeatedly updates a “world state”
  - Programmer must define what the “World” is
  - With a Data Definition!

```
;; A World is a non-negative integer
;; Interp: represents the y coordinate of a
ball in an animation
```

- Programmers specify “handler” functions to manipulate “World”
  - Render
  - Input handlers
  - World update

Big-bang code demo

Check-In Quiz 9/13

on gradescope

(due 1 minute before midnight)