# CS3:
## Introduction to Symbolic Programming

Lecture 2:
Introduction, and Conditionals

**Spring 2006**

**Nate Titterton**
**nate@berkeley.edu**

# Announcements

- **Nate's office hours:**
  - **Wednesday, 2 - 4**
  - **329 Soda**
- **Signup for card keys at 387 Soda Hall**
- **I'm not hearing about any book or reader supply problems.  Yes?**

# Any questions?

**Grading?**

**Working at home?**

# Schedule

| | | |
|---|---|---|
| 1 | Jan 16-20 | Lecture: <holiday><br>Lab: Introduction, words and sentences |
| 2 | Jan 23-27 | Lecture: Introduction, Conditionals<br>Lab: Conditionals |
| 3 | Jan 30-Feb 4 | Lecture: Case Studies<br>Lab: Work with Difference between Dates |
| 4 | Feb 6-10 | Lecture: Data abstraction in DbD<br>Lab: Miniproject 1 |
| 5 | Feb 13-17 | Lecture: Introduction to recursion<br>Lab: Recursion |

# Review

- **What is Scheme?**
  - A easy yet very powerful language
  - The "Listener" makes testing easy
- **Functions and functional programming**
- **Words and sentences**
  - Not usually part of scheme, but makes our early work more accessible

# Some programming

- **“first-two”**
  - takes a word, returns the first two letters (as a two-letter word)

- **“two-first”**
  - takes a sentence of two words, returns the first letter of each (as a two-letter word)

# A big idea

- **Data abstraction**
  - Constructors: procedures to make a piece of data
    - `word` and `sentence`
  - Selectors: procedures to return parts of that data piece
    - `first`, `butfirst`, etc.

# Some review

- **Quoting something means treating it *literally*:**
  - **you are interested in the thing follows, rather than what is named**
  - **Quoting is a shortcut to putting literal things right in your code. As your programs get bigger, you will do this less and less.**

**Quoting is something unique to Scheme (and similar language)**

# Coming up: conditionals

- **Conditionals allow programs to do different things depending on data values**
  - ***To make decisions***

- **"Intelligence" depends on this**

# Structure of conditionals

```
(if  <true? or false?>
     <do something if true>
     <do something if false>)
```

```
(define (smarty x)
   (if (odd? x)
      (se x '(is odd))
      (se x '(is even)))
      )
```

# *true? or false?*

- **We need Booleans: something that represents TRUE or FALSE to the computer**
  - `#t`
  - `#f`

  **- in practice, everything is true except `#f`**

  `false` **is true!**

  **(really,** `false` **is** `#t`**)**

# Predicates

- **Predicates are procedures that return `#t` or `#f`**
  - **by convention, their names end with a "?"**

```
odd?      (odd? 3) ➔ #t
even?     (even? 3) ➔ #f
vowel?    (vowel? 'a) ➔ #t
          (vowel? (first 'fred)) ➔ #f
sentence?     (sentence? 'fred) ➔ #f
```

# Coming up: testing

- **There is much more to programming than writing code**
  - ***Testing* is crucial, and an emphasis of this course**

  - **Analysis**
  - **Debugging**
  - **Maintenance.**
  - **"Design"**