Lecture 4:
"Difference Between Dates"
and
data abstraction
Announcements

• Nate's office hours *this week only*:
  - Thursday, 2-4, in 329 Soda
  - (Usually, they are Wed 2-4)
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture: Introduction, Conditionals</th>
<th>Lab: Conditionals</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Jan 23-27</td>
<td>Lecture: Case Studies</td>
<td>Reading: <strong>Difference between Dates</strong> Lab: Work with Difference between Dates</td>
</tr>
<tr>
<td>3</td>
<td>Jan 30-Feb 4</td>
<td>Lecture: Data abstraction in DbD</td>
<td>Lab: Miniproject I</td>
</tr>
<tr>
<td>4</td>
<td>Feb 6-10</td>
<td><strong>Lecture: HOLIDAY</strong></td>
<td>Lab: Recursion II</td>
</tr>
<tr>
<td>5</td>
<td>Feb 13-17</td>
<td>Lecture: Introduction to Recursion</td>
<td>Lab: Recursion</td>
</tr>
<tr>
<td>6</td>
<td>Feb 20-24</td>
<td>Lecture: <strong>Midterm 1</strong></td>
<td>Lab: Recursion III</td>
</tr>
<tr>
<td>7</td>
<td>Feb 27-Mar 3</td>
<td>Lecture: Midterm 1</td>
<td>Lab: Recursion III</td>
</tr>
</tbody>
</table>
How useful has the case study been?
Miniproject #1: this week

- You are to write century-day-span
  - Calculate the number of days between dates in (possibly) two different years

- Consider the central lesson of the case study: there are easier and harder ways to solve problems. Choose easier.
This is your first large program

- Use helper functions
- Test, and test some more.
- Reuse code that you have already written
- Add comments!
A Big Idea: *abstraction*

“The process of leaving out consideration of one or more properties of a complex object or process so as to attend to others.”
• Abstracting with a new function
  - (square x) instead of (* x x)
  - (third sent) instead of (first (bf (bf sent)))

• Abstracting a new datatype

  A datatype provides functionality necessary to store "something" important to the program
  - **Selectors**: to look at parts of the "something".
  - **Constructor**: to create a new "something".
  - **Tests (sometimes)**: to see whether you have a "something", or a "something else"
Data abstraction: words and sentences

- **Constructors**: procedures to make a piece of data
  - *word*
  - *sentence*

- **Selectors**: procedures to return parts of that data piece
  - *first*, *butfirst*, *etc.*
Benefits

• Why is "leaving out consideration of", or "not knowing about", a portion of the program a good thing?

• Consider two ways one can "understand a program":
  - Knowing what each function does
  - Knowing what the inputs are (can be), and what the outputs are (will be).
• Disregarding the "understanding" issue, why might it be a good idea to "modularize" your code?

(where modules are abstracted from each other)
Data abstraction in the DbD code

• How does the code separate out processing of the date-format from the logic that does the "real" work?

  - **Selectors**
    - month-name (takes a date)
    - date-in-month (takes a date)
    - ? month-number (takes a month name)

  - **Constructors?** **Tests?**