UNIT 2B
An Introduction to Programming
(for loops)
Announcements

• Office hours everyday of the week
• Academic integrity forms overdue!
• Hand in Problem Set 1 now
• Should be reading
  – EC Chapter 2, BB Chapter 2 pages 19-42

Always check the course Web page
Last Lecture

• Basic datatypes
• Variables
• Expressions
• Assignment statements
• Methods (functions)
This Lecture

• A control structure for iteration: “for loops”

• More Ruby practice
for Loop

for loop variable in start .. end do
  loop body
end

• The loop body is one or more instructions that you want to repeat.

• If \( start \leq end \), the for loop repeats the loop body \( end - start + 1 \) times.

• If \( start > end \), the entire loop is skipped.
for Loop Example

```plaintext
for i in 1..5 do
    print "hello world\n"
end

hello world
hello world
hello world
hello world
hello world
```
for Loop Example

for i in 1..5 do
  print i, "\n"
end

You can also use
  puts i instead of print i, "\n"
for Loop Example

```python
for i in 1..5 do
    print i
end

12345

for i in 1..5 do
    print i, " "
end

1 2 3 4 5
```

15110 Principles of Computing, Carnegie Mellon University
Assignment Statements

\[ \text{variable} = \text{expression} \]

The expression is evaluated and the result is stored in the variable, overwriting the previous contents of the variable.
### Assignment Statements

<table>
<thead>
<tr>
<th>statement</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 150$</td>
<td>150</td>
<td>?</td>
</tr>
<tr>
<td>$y = x \times 10$</td>
<td>150</td>
<td>1500</td>
</tr>
<tr>
<td>$y = y + 1$</td>
<td>150</td>
<td>1501</td>
</tr>
<tr>
<td>$x = x + y$</td>
<td>1651</td>
<td>1501</td>
</tr>
</tbody>
</table>
A function using a for loop

```python
def sum()  
    # sums the first 3 positive integers  
    sum = 0  
    for i in 1..3 do  
        sum = sum + i  
    end  
    return sum  
end
```

<table>
<thead>
<tr>
<th>i</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize sum</td>
<td>?</td>
</tr>
<tr>
<td>iteration 1</td>
<td>1</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>2</td>
</tr>
<tr>
<td>iteration 3</td>
<td>3</td>
</tr>
</tbody>
</table>
Danger!

```python
for i in 0..6 do
    i = i + 2
    print i, " "
end
```

2 3 4 5 6 7 8

If you modify the loop variable inside of the `for` loop, the loop will reset the loop variable to its next expected value in the next iteration.

Programming suggestion: Do NOT modify the loop variable inside a `for` loop.
Generalizing our solution

```python
def sum(n):
    # sums the first n positive integers
    sum = 0
    for i in 1..n do
        sum = sum + i
    end
    return sum
end
```

sum(6)  =>  21
sum(100) => 5050
sum(15110) => 114163605
def compute_sick(n)
    # computes total sick after n days
    newly_sick = 1
    total_sick = 1
    for day in 2..n do
        # each iteration represents one day
        newly_sick = newly_sick * 2
        total_sick = total_sick + newly_sick
    end
    return total_sick
end

Each newly infected person infects 2 people the next day.
An epidemic (cont’d)

compute_sick(1)  =>  1  compute_sick(17)  =>  131071
compute_sick(2)  =>  3  compute_sick(18)  =>  262143
compute_sick(3)  =>  7  compute_sick(19)  =>  524287
compute_sick(4)  =>  15 compute_sick(20)  =>  1048575
compute_sick(5)  =>  31 compute_sick(21)  =>  2097151
compute_sick(6)  =>  63
compute_sick(7)  =>  127
compute_sick(8)  =>  255
compute_sick(9)  =>  511
compute_sick(10) =>  1023
compute_sick(11) =>  2047
compute_sick(12) =>  4095
compute_sick(13) =>  8191
compute_sick(14) =>  16383
compute_sick(15) =>  32767
compute_sick(16) =>  65535

In just three weeks, over 2 million people are sick! (This is what Blown To Bits means by exponential growth. We will see important computational problems that get exponentially “harder” as the problems gets bigger.)
def countdown()  
    for i in 1..10 do  
        print 11-i  
        print " "  
        sleep 1  # pauses for 1 sec.  
    end  
end  

countdown()  
=> 10 9 8 7 6 5 4 3 2 1
Next Week

• New concept: algorithm
• New control structures in Ruby
  – While loops
  – Conditionals