UNIT 2B
An Introduction to Programming

Announcements

- Tutoring help on Mondays 8:30-11:00 pm in the Mudge Reading Room
- Extra help session Fridays 12:00-2:00 pm in GHC 4122
- Academic integrity forms

Always check the course Web page and Piazza
Last Lecture

- Basic datatypes
- Variables
- Expressions
- Assignment statements
- Methods (functions)
- Modules such as Math

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 ≤ end, the for loop repeats the loop body \( \text{end} - \text{start} + 1 \) times.

• If start > end, the entire loop is skipped.

for Loop Example

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
    print "\n"
end

1
2
3
4
5

for Loop Example

for i in 1..5 do
    print i
end

12345

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

1 2 3 4 5
for Loop Example

for i in 1..10 do
  print i*2
  print " "
end

2 4 6 8 10 12 14 16 18 20

Assignment Statements

variable = 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 * 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>

Danger!

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

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.
A function using a for loop

```python
def summation()
    # 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>?</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Generalizing our solution

```python
def summation(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
```
An epidemic

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 get 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