Wrapping up

- We'll talk just a bit more about code blocks
- Then cover exceptions, command line arguments, and some other Ruby details

More Examples of Code Block Usage

- Sum up the elements of an array

```
a = [1,2,3,4,5]
sum = 0
a.each { |x| sum = sum + x }
printf("sum is %d\n", sum)
```

- Print out each segment of the string as divided up by commas (commas are printed trailing each segment)
  - Can use any delimiter

```
s = "Student,Sally,099112233,A"
s.split(',').each { |x| puts x }
```

("delimiter" = symbol used to denote boundaries)

Yet More Examples of Code Blocks

```
3.times { puts "hello"; puts "goodbye" }
5.upto(10) { |x| puts(x + 1) }
[1,2,3,4,5].find { |y| y % 2 == 0 }
[5,4,3].collect { |x| -x }
```

- n.times runs code block n times
- n.upto(m) runs code block for integers n..m
- a.find returns first element x of array such that the block returns true for x
- a.collect applies block to each element of array and returns new array (a.collect! modifies the original)

Still Another Example of Code Blocks

```
File.open("test.txt", "r") do |f|
  f.readlines.each { |line| puts line }
end
```

- open method takes code block with file argument
  - File automatically closed after block executed
- readlines reads all lines from a file and returns an array of the lines read
  - Use each to iterate

Using Yield To Call Code Blocks

- Any method can be called with a code block
  - Inside the method, the block is called with yield
- After the code block completes
  - Control returns to the caller after the yield instruction

```
def countx(x)
  for i in (1..x)
     puts i
     yield
  end
end

countx(4) { puts "foo" }
```

```
1
foo
2
foo
3
foo
4
foo
```

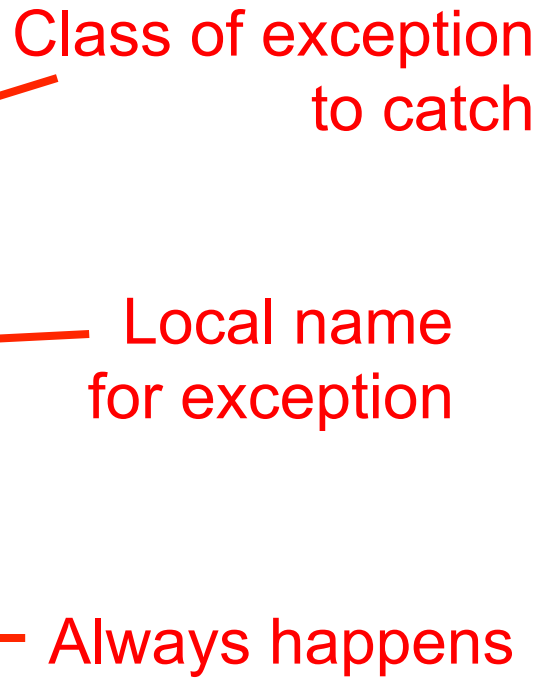
So What Are Code Blocks?

- A code block is just a special kind of method
  - { |y| x = y + 1; puts x } is almost the same as
  - def m(y) x = y + 1; puts x end
- The each method takes a code block as an argument
  - This is called higher-order programming
    - In other words, methods take other methods as arguments
    - We'll see a lot more of this in OCaml
- We'll see other library classes with each methods
  - And other methods that take code blocks as arguments
  - As we saw, your methods can use code blocks too!

Exceptions

- Use begin...rescue...ensure...end
  - Like try...catch...finally in Java

```
begin
  f = File.open("test.txt", "r")
  while !f.eof
    line = f.readline
    puts line
  end
rescue Exception => e
  puts "Exception:" + e.to_s +
   " (class " + e.class.to_s + ")"
ensure
  f.close if f != nil
end
```



Command Line Arguments

- Stored in predefined array variable $*
  - Can refer to as predefined global constant ARGV

- Example
  - If
    - Invoke test.rb as “ruby test.rb a b c”
  - Then
    - ARGV[0] = “a”
    - ARGV[1] = “b”
    - ARGV[2] = “c”

Comparisons

- Sorting requires ability to compare two values
- Ruby comparison method <=>
  - -1 = less
  - 0 = equals
  - +1 = greater
- Examples
  - 3 <=> 4 returns -1
  - 4 <=> 3 returns +1
  - 3 <=> 3 returns 0

Sorting

- Two ways to sort an Array
  - Default sort (puts values in ascending order)
    - [2,5,1,3,4].sort # returns [1,2,3,4,5]
  - Custom sort (based on value returned by code block}
    - [2,5,1,3,4].sort { |x,y| y <=> x } # returns [5,4,3,2,1]
    - Where -1 = less, 0 = equals, +1 = greater
    - Code block return value used for comparisons

Ruby Summary

- Interpreted
- Implicit declarations
- Dynamically typed
- Built-in regular expressions
- Easy string manipulation
- Object-oriented
  - Everything (!) is an object
- Code blocks
  - Easy higher-order programming!
  - Get ready for a lot more of this...

Makes it quick to write small programs

Hallmark of scripting languages

Other Scripting Languages

- Perl and Python are also popular scripting languages
  - Also are interpreted, use implicit declarations and dynamic typing, have easy string manipulation
  - Both include optional "compilation" for speed of loading/execution
- Will look fairly familiar to you after Ruby
  - Lots of the same core ideas
  - All three have their proponents and detractors
  - Use whichever language you personally prefer

Example Perl Program

```
#!/usr/bin/perl
foreach (split//, $ARGV[0])) {
  if ($G{$_}) {
    $RE .= "\\" . $G{$_};
  } else {
    $RE .= $N ? "(?!\\" .
  join("|\\",values(%G)) . ')(\w)' : '(\w)';
    $G{$_} = ++$N;
}  }
```

Example Python Program

```
#!/usr/bin/python
import re
list = ("deep", "deer", "duck")
x = re.compile("^\S{3,5}.[aeiou]")
for i in list:
  if re.match(x, i):
    print I
  else:
    print
```