CSE 413 Autumn 2008

Ruby Containers, Iterators, and Blocks

Containers in Ruby

- Ruby has general, easy-to-use container classes, like most scripting languages
- Two major kinds
  - Arrays: ordered by position
  - Hashes: collections of <key, value> pairs
    - Often known as associative arrays, maps, or dictionaries
    - Unordered

Ruby Arrays

- Instances of class Array
- Create with an array literal, or Array.new
  - words = [“how”, “now”, “brown”, “cow”]
  - stuff = [“thing”, 413, nil]
  - seq = Array.new
- Indexed with [ ] operator, 0-origin; negative indices count from right
  - words[0] stuff[2] words[-2]
  - seq[1] = “something”

Ruby Hashes

- Instances of class Hash
- Create with an hash literal, or Hash.new
  - pets = { “spot” => “dog”, “puff” => “cat” }
  - tbl = Array.new
- Indexed with [ ] operator
  - pets[“puff”] pets[“fido”]
  - pets[“cheeta”] = “monkey”
  - (Can use almost anything as key type; can use anything as element type)

Containers and Iterators

- All containers respond to the message “each”, executing a block of code for each item in the container
  - words.each { puts “another word” }
  - words.each { | w | puts w }

Blocks

- A block is a sequence of statements surrounded by { … } or do … end
- Blocks must appear immediately following the method call that executes them, on the same line
- Blocks may have 1 or more parameters at the beginning surrounded by | … |
  - Initialized by the method that runs the block

Blocks as Closures

- Blocks can access variables in surrounding scopes
  - all_words
    words.each { | w | all_words = all_words + w + “ ” }
  - These are almost, but not quite, first-class closures as in Scheme (some differences in scope rules)

More Block Uses

- Besides iterating through containers, blocks are used in many other contexts
  - 3.times { puts “hello” }
  - n = 0
    100.times { | k | n += k }
    puts “sum of 0 + … + 99 is ” + n

  - We’ll see more examples of blocks as well as how to write code that uses them later