UW Ruby Programming 110

Winter 2015

Michael Cohen

Lecture 8

Feb 26, 2015

Lecture 8

1. Enumerable
2. ActiveSupport
3. Assignment 8
4. Final Project

Section 1

Enumerable

Section 1: Enumerable

Basic Implementation

```
each_with_index
find
find_all
map
reduce
```

each with index

```
def each_with_index
  index = 0
  each do |item|
    yield item, index
    index += 1
  end
end
```

Section 1: Enumerable

find

```
def find
   each do |item|
      return item  if yield item
   end
   nil
end
```

find_all

```
def find_all
   found_items = []
   each do |item|
      found_items << item  if yield item
   end
   found_items
end
```

map

```
def map
   items = []
   each do |item|
      items << yield item
   end
   items
end
```

map

```
def reduce(acc)
  each do |item|
    acc = yield acc, item
  end
  acc
end
```

Section 2

ActiveSupport

What is ActiveSupport?

ActiveSupport is a part of Rails that extends built-in classes (Object, Array, Hash, String, etc.)

Section 2: ActiveSupport

Installation & Usage

installation:

```
gem install activesupport
```

usage:

```
require 'active_support/all'
```

Section 2: ActiveSupport

Object

```
obj.blank?
obj.present?
```

Section 2: ActiveSupport

Object - try

```
# without try:
unless @number.nil?
   @number.next
end

# with try:
@number.try(:next)
```

Section 2: ActiveSupport

String

```
# remove:
"Hello World".remove(/Hello /)  # => "World"unless @number.nil?

# squish:
" \n  foo\n\r \t bar \n".squish # => "foo bar"

# truncate:
"Oh dear! Oh dear! I shall be late!".truncate(20)  # => "Oh dear! Oh dear!..."
```

Section 2: ActiveSupport

String

```
# starts_with?:
"foo".starts_with?("f") # => true

# ends_with?:
"foo".ends_with?("o")   # => true
```

Section 2: ActiveSupport

String - Access

```
# at(position):
"hello".at(0)    # => "h"
"hello".at(4)    # => "o"
"hello".at(-1)   # => "o"
"hello".at(10)   # => nil
```

Section 2: ActiveSupport

String - Access

```
# from(position):
"hello".from(0)    # => "hello"
"hello".from(2)    # => "llo"
"hello".from(-2)   # => "lo"
"hello".from(10)   # => "" if < 1.9, nil in 1.9
```

Section 2: ActiveSupport

String - Access

```
# to(position):
"hello".to(0)    # => "h"
"hello".to(2)    # => "hel"
"hello".to(-2)   # => "hell"
"hello".to(10)   # => "hello"
```

Section 2: ActiveSupport

String - Inflections

```
# pluralize:
"table".pluralize        # => "tables"
"ruby".pluralize         # => "rubies"
"equipment".pluralize    # => "equipment"

"dude".pluralize(0)      # => "dudes"
"dude".pluralize(1)      # => "dude"
"dude".pluralize(2)      # => "dudes"
```

Section 2: ActiveSupport

String - Inflections

```
# singularize:
"tables".singularize      # => "table"
"rubies".singularize      # => "ruby"
"equipment".singularize # => "equipment"
```

Section 2: ActiveSupport

String - Inflections

```
# camelize:
"product".camelize     # => "Product"
"admin_user".camelize  # => "AdminUser"
```

Section 2: ActiveSupport

String - Inflections

```
# underscore:
"Product".underscore    # => "product"
"AdminUser".underscore # => "admin_user"
```

Section 2: ActiveSupport

String - Inflections

```
# titleize:
"alice in wonderland".titleize # => "Alice In Wonderland"
"fermat's enigma".titleize     # => "Fermat's Enigma"
```

Section 2: ActiveSupport

String - Inflections

```
# dasherize:
"name".dasherize                 # => "name"
"contact_data".dasherize # => "contact-data"
```

Section 2: ActiveSupport

String - Inflections

```
# tableize:
"Person".tableize         # => "people"
"Invoice".tableize        # => "invoices"
"InvoiceLine".tableize    # => "invoice_lines"
```

Section 2: ActiveSupport

String - Inflections

```
# classify:
"people".classify            # => "Person"
"invoices".classify          # => "Invoice"
"invoice_lines".classify # => "InvoiceLine"
```

Section 2: ActiveSupport

String - Inflections

```
# humanize:
"name".humanize                                 # => "Name"
"author_id".humanize                            # => "Author"
"author_id".humanize(capitalize: false)         # => "author"
"comments_count".humanize                       # => "Comments count"
"_id".humanize                                  # => "Id"
```

Section 2: ActiveSupport

String - Conversions

```
# to_date, to_time, to_datetime:
"2010-07-27".to_date                  # => Tue, 27 Jul 2010
"2010-07-27 23:37:00".to_time         # => Tue Jul 27 23:37:00 UTC 2010
"2010-07-27 23:37:00".to_datetime # => Tue, 27 Jul 2010 23:37:00 +0000
```

Numeric - Bytes

```
# bytes, kilobytes, megabytes, gigabytes
# terabytes, petabytes, exabytes
2.kilobytes    # => 2048
3.megabytes    # => 3145728
3.5.gigabytes  # => 3758096384
-4.exabytes    # => -4611686018427387904
```

Section 2: ActiveSupport

Numeric - Time

```
# equivalent to Time.current.advance(months: 1)
1.month.from_now

# equivalent to Time.current.advance(years: 2)
2.years.from_now

# equivalent to Time.current.advance(months: 4, years: 5)
(4.months + 5.years).from_now
```

Section 2: ActiveSupport

Numeric - Formatting

```
5551234.to_s(:phone)       # => 555-1234
1235551234.to_s(:phone)    # => 123-555-1234

1235551234.to_s(:phone, area_code: true)
# => (123) 555-1234

1235551234.to_s(:phone, delimiter: " ")
# => 123 555 1234

1235551234.to_s(:phone, area_code: true, extension: 555)
# => (123) 555-1234 x 555

1235551234.to_s(:phone, country_code: 1)
# => +1-123-555-1234
```

Section 2: ActiveSupport

Numeric - Formatting

```
1234567890.50.to_s(:currency)                     # => $1,234,567,890.50
1234567890.506.to_s(:currency)                    # => $1,234,567,890.51
1234567890.506.to_s(:currency, precision: 3)      # => $1,234,567,890.506
```

Section 2: ActiveSupport

Numeric - Formatting

```ruby
100.to_s(:percentage)
# => 100.000%

100.to_s(:percentage, precision: 0)
# => 100%

1000.to_s(:percentage, delimiter: '.', separator: ',')
# => 1.000,000%

302.24398923423.to_s(:percentage, precision: 5)
# => 302.24399%
```

Section 2: ActiveSupport

Numeric - Formatting

```
12345678.to_s(:delimited)                       # => 12,345,678
12345678.05.to_s(:delimited)                    # => 12,345,678.05
12345678.to_s(:delimited, delimiter: ".")       # => 12.345.678
12345678.to_s(:delimited, delimiter: ",")       # => 12,345,678
12345678.05.to_s(:delimited, separator: " ")    # => 12,345,678 05
```

Section 2: ActiveSupport

Numeric - Formatting

```
111.2345.to_s(:rounded)                          # => 111.235
111.2345.to_s(:rounded, precision: 2)            # => 111.23
13.to_s(:rounded, precision: 5)                  # => 13.00000
389.32314.to_s(:rounded, precision: 0)           # => 389
111.2345.to_s(:rounded, significant: true)       # => 111
```

Numeric - Formatting

```
123.to_s(:human_size)                  # => 123 Bytes
1234.to_s(:human_size)                 # => 1.21 KB
12345.to_s(:human_size)                # => 12.1 KB
1234567.to_s(:human_size)              # => 1.18 MB
1234567890.to_s(:human_size)           # => 1.15 GB
1234567890123.to_s(:human_size)        # => 1.12 TB
```

Section 2: ActiveSupport

Numeric - Formatting

```
123.to_s(:human)                  # => "123"
1234.to_s(:human)                 # => "1.23 Thousand"
12345.to_s(:human)                # => "12.3 Thousand"
1234567.to_s(:human)              # => "1.23 Million"
1234567890.to_s(:human)           # => "1.23 Billion"
1234567890123.to_s(:human)        # => "1.23 Trillion"
1234567890123456.to_s(:human)     # => "1.23 Quadrillion"
```

Integer

```
# multiple_of? :
2.multiple_of?(1) # => true
1.multiple_of?(2) # => false
```

Integer

```
# ordinal:
1.ordinal     # => "st"
2.ordinal     # => "nd"
53.ordinal    # => "rd"
2009.ordinal  # => "th"
-21.ordinal   # => "st"
-134.ordinal  # => "th"
```

Integer

```
# ordinalize:
1.ordinalize     # => "1st"
2.ordinalize     # => "2nd"
53.ordinalize    # => "53rd"
2009.ordinalize  # => "2009th"
-21.ordinalize   # => "-21st"
-134.ordinalize  # => "-134th"
```

Section 2: ActiveSupport

Enumerable

```
# sum:
[1, 2, 3].sum # => 6
(1..100).sum  # => 5050

[[1, 2], [2, 3], [3, 4]].sum     # => [1, 2, 2, 3, 3, 4]
%w(foo bar baz).sum              # => "foobarbaz"
{a: 1, b: 2, c: 3}.sum           # => [:b, 2, :c, 3, :a, 1]
```

Section 2: ActiveSupport

Array - Adding

```
# prepend (unshift alias):
%w(a b c d).prepend('e')   # => %w(e a b c d)
[].prepend(10)             # => [10]

# append (<< alias):
%w(a b c d).append('e')    # => %w(a b c d e)
[].append([1,2])           # => [[1,2]]
```

Section 2: ActiveSupport

Array - Conversions

```
# to_sentence:
%w().to_sentence                    # => ""
%w(Earth).to_sentence               # => "Earth"
%w(Earth Wind).to_sentence          # => "Earth and Wind"
%w(Earth Wind Fire).to_sentence # => "Earth, Wind, and Fire"
```

Section 2: ActiveSupport

Array - Wrapping

```
# wrap:
Array.wrap(nil)       # => []
Array.wrap([1, 2, 3]) # => [1, 2, 3]
Array.wrap(0)         # => [0]
```

Section 2: ActiveSupport

Array - Grouping

```
[1, 2, 3].in_groups_of(2) # => [[1, 2], [3, nil]]

[1, 2, 3].in_groups_of(2, 0) # => [[1, 2], [3, 0]]

[1, 2, 3].in_groups_of(2, false) # => [[1, 2], [3]]
```

Section 2: ActiveSupport

Hash - Conversions

```
# to_xml:
{"foo" => 1, "bar" => 2}.to_xml
# =>
# <?xml version="1.0" encoding="UTF-8"?>
# <hash>
#   <foo type="integer">1</foo>
#   <bar type="integer">2</bar>
# </hash>
```

Section 3

Assignment #8

Section 3: Assignment #8

Problem 1: Roman Numerals

```
# implement conversion between integers and roman numerals
# validate using MiniTest unit tests

module Assignment08
  class RomanNumeral
    def initialize(i)
      # your implementation here
    end

    def to_s
      # your implementation here
    end

    def to_i
      # your implementation here
    end
  end
end
```

Section 3: Assignment #8

Problem 1: Roman Numerals

```
# expected results:
RomanNumeral.new(1).to_s    # => 'I'
RomanNumeral.new(2).to_s    # => 'II'
RomanNumeral.new(3).to_s    # => 'III'
RomanNumeral.new(4).to_s    # => 'IV'
RomanNumeral.new(5).to_s    # => 'V'
RomanNumeral.new(6).to_s    # => 'VI'
RomanNumeral.new(9).to_s    # => 'IX'
RomanNumeral.new(10).to_s   # => 'X'
RomanNumeral.new(19).to_s   # => 'XIX'
RomanNumeral.new(32).to_s   # => 'XXXII'
RomanNumeral.new(51).to_s   # => 'LI'
```

Section 3: Assignment #8

Problem 2: Golden Ratio

```
# Golden Ratio is ratio between consecutive Fibonacci numbers
# calculate the golden ratio up to specified precision
# validate using MiniTest unit tests

module Assignment08
  def golden_ratio(precision)
    # your implementation here
  end
end

# expected results:
golden_ratio(2)  # => 1.62
golden_ratio(5)  # => 1.61803
golden_ratio(8)  # => 1.61803399
```

Section 4

Final Project

Final Project

Game of Life

The Game of Life is a simplified model of evolution and natural selection invented by the mathematician James Conway.

`http://en.wikipedia.org/wiki/Conway's_Game_of_Life`

Final Project: Game of Life

Rules

You have a grid of cells in 2 dimensions.
Each cell has 8 neighbors:
- top, right, bottom, left
- top-left, top-right, bottom-right, bottom-left
Each cell has 2 possible states: alive or dead.

Final Project: Game of Life

Rules

if a cell is alive and
- has fewer than 2 live neighbors, it dies
- has more than 3 live neighbors, it dies
- has 2 or 3 live neighbors, it lives to next generation

Final Project: Game of Life

Rules

if cell is dead and
- has exactly 3 live neighbors, it becomes a live cell

Final Project: Game of Life

Rules

edges of board:
- pretend the board is folded onto itself
- the edges touch each other

Final Project: Game of Life

Suggested Implementation

```
class GameOfLife
  def initialize(size)
    # randomly initialize the board
  end
  def evolve
    # apply rules to each cell and generate new state
  end
  def render
    # render the current state of the board
  end
  def run(num_generations)
    # evolve and render for num_generations
  end
end
```

Final Project: Game of Life

Tests

You must have MiniTest unit tests for your class to validate your implementation.

You might need to add methods or change the method signatures to enable testing.

Final Project: Game of Life

Rendering

You choose how you want to render the current state of the board. ASCII? HTML? Something else?

Bonus: DSL

- Create a DSL that represents a state of the game.
- Your render method can then be formatted as the DSL, so that you can round-trip between the textual DSL representation and the running instance.