UW Ruby Programming 110

Winter 2015

Michael Cohen

Lecture 3

Jan 22, 2015

Assignment #2

Assignment #2

Problem 1

```
def to_sentence(ary)
   if ary.length == 0
      []
   elsif ary.length == 1
      ary[0]
   else
      last = ary.pop
      "#{ary.join(", ")} and #{last}"
   end
end
```

Assignment #2

Problem 2

```ruby
def mean(ary)
  sum = ary.reduce(0) {|x, acc| acc + x}
  sum.to_f / ary.length
end

def median(ary)
  sorted_ary = ary.sort
  len = sorted_ary.length
  mid_index = len/2
  if len.odd?
    sorted_ary[mid_index]
  else
    mid_lo = sorted_ary[mid_index]
    mid_hi = sorted_ary[mid_index+1]
    (mid_lo + mid_hi)/2.0
  end
end
```

Assignment #2

Problem 3

```
def pluck(ary, key)
  ary.map {|item| item[key]}
end
```

Lecture 2

1. Methods & Blocks
2. Classes & Objects
3. Assignments

Section 1

Methods & Blocks

Section 1: Methods & Blocks

Arguments

```ruby
def myNewMethod(arg1, arg2, arg3)      # 3 arguments
  # Code for the method would go here
end

def myOtherNewMethod                   # No arguments
  # Code for the method would go here
end
```

Section 1: Methods & Blocks

Default Values

```
def coolDude(arg1="Miles", arg2="Coltrane", arg3="Roach")
  "#{arg1}, #{arg2}, #{arg3}."
end
coolDude                          #=> "Miles, Coltrane, Roach."
coolDude("Bart")                  #=> "Bart, Coltrane, Roach."
coolDude("Bart", "Elwood")   #=> "Bart, Elwood, Roach."
coolDude("Bart", "Elwood", "Linus")   #=> "Bart, Elwood, Linus."
```

Section 1: Methods & Blocks

Variable-Length Argument Lists

```
def varargs(arg1, *rest)
  "Got #{arg1} and #{rest.join(', ')}"
end
varargs("one")                      #=> "Got one and "
varargs("one", "two")               #=> "Got one and two"
varargs "one", "two", "three"  #=> "Got one and two, three"
```

Hash arguments

```ruby
# only hash args:
def stuff(options)
  # ...
end

stuff key1: "a", key2: "b"
```

Hash arguments

```
# regular args and hash args:
def stuff2(first, options)
   # ...
end

stuff2 "hello", key1: "a", key2: "b"
```

Hash arguments

```
# optional hash args:
def stuff3(first, options={})
  # ...
end

stuff3 "hello"
```

Return

```
def work(arg1, arg2)
   if arg1.nil?
      # do stuff
      return false
   end

   # continue on with other work.
   true
end
```

Blocks

```ruby
def takeBlock(p1)
  if block_given?
    yield(p1)
  else
    p1
  end
end

takeBlock "no block"                #=> "no block"
takeBlock "" { puts "block" }       #=> "block"
```

Blocks

```ruby
def takeBlock(p1, &b)
  if block_given?
    b.call p1
  else
    p1
  end
end

takeBlock "no block"                #=> "no block"
takeBlock "" { puts "block" }       #=> "block"
```

Blocks

```
def render_html(title)
<<HTML
      <!doctype html>
      <html>
        #{render_head title}
        #{yield title}
      </html>
HTML
end
```

Blocks

```
def render_body(title)
<<BODY
   <body>
      <h1>#{title}</h1>
      #{yield title}
   </body>
BODY
end
```

Blocks

```
render_html("") do |title|
  render_body(title) do
    render records records
  end
end
```

Section 2

Objects & Classes

Objects

- Everything is an object
- Every object is an instance of a class
- Method invocation is sending a message to an object

to_s

```
xs = [1,2,3]
h = {key1: "hello", key2: "world"}

xs.to_s      #=> "[1, 2, 3]"
h.to_s       #=> "{:key1=>\"hello\", :key2=>\"world\"}"
```

Section 2: Objects & Classes

class, is*a?*, *instance*of?

```
xs = []

xs.class                    #=> Array
xs.is_a? Array              #=> true
xs.is_a? Object             #=> true
xs.is_a? Hash               #=> false

xs.instance_of? Array       #=> true
xs.instance_of? Object      #=> false
xs.instance_of? Hash        #=> false
```

Classes are Objects

```
Array.is_a? Object      #=> true
Array.class             #=> Class

Hash.is_a?  Object      #=> true
Hash.class              #=> Class

Class.is_a? Object      #=> true
Class.class             #=> Class
```

Section 2: Objects & Classes

methods

```
Array.methods     #=> array of methods supported by Array

Array.methods.include? :slice    #=> true
Array.methods.include? :funny    #=> false
```

send

```
xs = [1,2,3]

xs.length          #=> 3
```

```
xs.send :length    #=> 3
```

implicit self

```ruby
def method1(arg1)
  # ...
end
```

```ruby
# these four are equivalent:
method1 "hello"
self.method1 "hello"
send :method1, "hello"
self.send :method1, "hello"
```

respond_to?

```
xs = [1,2,3]
h = {key1: "hello", key2: "world"}

xs.respond_to? :length      #=> true
xs.respond_to? :keys        #=> false
xs.respond_to? :slice       #=> true

h.respond_to? :length       #=> true
h.respond_to? :keys         #=> true
h.respond_to? :slice        #=> false
```

Defining a Class

```
class MyClass
   def method1
      puts "hello"
   end
end

my_instance = MyClass.new
```

Initialize

```
class MyClass
   def initialize(height, width)
      # so stuff with height, width
   end
end

my_instance = MyClass.new 10, 20
```

Section 2: Objects & Classes

Instance Variables

```ruby
class MyClass
  def initialize(height, width)
    @height = height
    @width = width
  end
  def describe
    puts "height: #{@height}, width: #{@width}"
  end
end

my_instance = MyClass.new 10, 20

my_instance.describe        #=> height: 10, width: 20
```

Section 2: Objects & Classes

Getters

```ruby
class MyClass
  def initialize(height, width)
    @height = height
    @width = width
  end

  # getters:
  def height; @height; end
  def width;  @width;  end
end

my_instance = MyClass.new 10, 20
my_instance.height  #=> 10
```

Section 2: Objects & Classes

Setters

```ruby
class MyClass
  # setters:
  def height=(new_height)
    @height = new_height
  end
  def width=(new_width)
    @width = new_width
  end
end

obj = MyClass.new 10, 20
obj.height                  #=> 10
obj.height = 40
obj.height                  #=> 40
```

Attr

```
class MyClass
   attr :height, :width

   def initialize(height, width)
      @height = height
      @width = width
   end
end
```

Attr variations

```
class MyClass
  attr_reader :height
  attr_writer :width

  def initialize(height, width)
    @height = height
    @width = width
  end
end
```

Access control

```
class MyClass
    def method1    # default is 'public'
        #...
    end
end
```

Access control: protected

```
class MyClass
  protected          # subsequent methods will be 'protected'
    def method2      # will be 'protected'
      #...
    end
end
```

Access control: private

```ruby
class MyClass
  private             # subsequent methods will be 'private'
    def method3       # will be 'private'
      #...
    end

  public              # subsequent methods will be 'public'
    def method4       # and this will be 'public'
      #...
    end
end
```

Section 2: Objects & Classes

Inheritance

```
class Vehicle
  attr :color, :num_wheels, :num_doors
  def initialize(color, num_wheels, num_doors)
    @color = color
    @num_wheels = num_wheels
    @num_doors = num_doors
  end

  def has_engine?
    false
  end

  def has_doors?
    num_doors > 0
  end
end
```

Inheritance

```
class Bicycle < Vehicle
  def initialize(color)
    super color, 2, 0
  end
end
```

Inheritance

```
class MotorizedVehicle < Vehicle
  def initialize(color, num_wheels, num_doors, engine)
    super color, num_wheels, num_doors
    @engine = engine
  end

  def has_engine?
    @engine != nil
  end
end
```

Inheritance

```
class Truck < MotorizedVehicle
  def initialize(color, engine)
    super color, 4, 2, engine
  end
end

class Car < Vehicle
  def initialize(color, engine)
    super color, 4, 4, engine
  end
end
```

Section 2: Objects & Classes

Inheritance

```
class Vehicle
  def max_speed
    0
  end
end
```

```
class Truck < MotorizedVehicle
  def max_speed
    60
  end
end

class Car < MotorizedVehicle
  def max_speed
    100
  end
end
```

Class variables

```
class Vehicle
  @@num_vehicles_created = 0
  @@last_vehicle_created = nil
  def initialize(color, num_wheels, num_doors)
    @color = color
    @num_wheels = num_wheels
    @num_doors = num_doors
    @@num_vehicles_created += 1
    @@last_vehicle_created = self
  end
end
```

Class methods

```ruby
class Vehicle
  @@num_vehicles_created = 0
  @@last_vehicle_created = nil

  def Vehicle.last
    @@last_vehicle_created
  end

  def self.num_created
    @@num_vehicles_created
  end
end
```

Section 2: Objects & Classes

Setter vs local variable

```ruby
class Vehicle
  attr :miles_driven
  def initialize
    # ...
    @miles_driven = 0
    # ...
  end

  def drive(num_miles)
    miles_driven += num_miles       # WARNING: this probably isn't what you intended

    self.miles_driven += num_miles # this is correct
  end
end
```

Section 2: Objects & Classes

Objects vs Hashes

```ruby
# using Hashes:
def create_address(street, city, state, zip)
  {street: street, city: city, state: state, zip: zip}
end

addr = {street: "123 Main St", city: "Seattle", state: "WA", zip: 98122}
```

Section 2: Objects & Classes

Objects vs Hashes

```ruby
# using Classes:
class Address
  attr :street, :city, :state, :zip

  def initialize(addr_info)
    @street = addr_info[:street]
    @city   = addr_info[:city]
    @state  = addr_info[:state]
    @zip    = addr_info[:zip]
  end
end

addr = Address.new street: "123 Main St", city: "Seattle", state: "WA", zip: 98122
```

Open Classes

```ruby
class Address
  def initialize(addr_info)
    @street = addr_info[:street]
    @city   = addr_info[:city]
    @state  = addr_info[:state]
    @zip    = addr_info[:zip]
  end
end

class Address
  attr :street, :city, :state, :zip
end
```

Open Classes

```ruby
def pluck(ary, key)
  ary.map {|item| item[key]}
end
```

```ruby
# add pluck to Array:
class Array
  def pluck(key)
    self.map {|item| item[key]}
  end
end
```

Section 2: Objects & Classes

implicit self

```
class Array
  def pluck(key)
    map {|item| item[key]}   # self is implicit
  end
end
```

respond_to?

```
class Array
  def pluck(key)
    map do |item|
      if item.respond_to? key
        item.send key
      else
        item[key]
      end
    end
  end
end
```

to_s

```
class Address
  def to_s
    "#{street}, #{city}, #{state} #{zip}"
  end
end
```

Class body is just code

```
DEBUG = true
class Vehicle
  def to_s
    if DEBUG
      "#{street}, #{city}, #{state} #{zip}"
    else
      ""
    end
  end
end
```

Class body

```
DEBUG = true
class Vehicle
  if DEBUG
    def to_s
      "#{street}, #{city}, #{state} #{zip}"
    end
  else
    def to_s
      ""
    end
  end
end
```

Section 3

Assignments

Section 3: Assignments

Problem 1 - re-implement titleize, palindrome?

```
# re-implement titleize and palindrome? as methods on String

"hEllo WORLD".titleize                                  #=> "Hello World"
"gooDbye CRUel wORLD".titleize                          #=> "Goodbye Cruel World"
```

```
"abba".palindrome?                                      #=> true
"aBbA".palindrome?                                      #=> true
"abb".palindrome?                                       #=> false

"Able was I ere I saw elba".palindrome?                 #=> true
"A man, a plan, a canal, Panama".palindrome?            #=> true
```

Section 3: Assignments

Problem 2 - re-implement mean, median, to_sentence

```
# re-implement mean, median, to_sentence as methods on Array

# Your method should generate the following results:
[1, 2, 3].mean      #=> 2
[1, 1, 4].mean      #=> 2

[1, 2, 3].median    #=> 2
[1, 1, 4].median    #=> 1

[].to_sentence                          #=> ""
["john"].to_sentence                    #=> "john"
["john", "paul"].to_sentence            #=> "john and paul"
[1, "paul", 3, "ringo"].to_sentence     #=> "1, paul, 3 and ringo"
```

Section 3: Assignments

Problem 3 - re-implement bank statement

```
# re-implement bank statement from Assignment 2

# instead of using hashes, create classes to represent:
# - BankAccount
# - Transaction
# - DepositTransaction
# - WithdrawalTransaction

# use blocks for your HTML rendering code
```