Date: 25-June-2016

• RVM

• String interpolation “” vs ‘’

• %W, %w, %Q etc.

• Class variables, instance variable, freeze, constants

• ! (bang operator)

• Splat operator (\*, \*\*)

• Duck typing

• String vs symbols (garbage collection)

• Hash default value

• && vs and

• sub, gsub, strip, compact

• Blocks

• Ranges

• Protected vs private

• each vs for\_each

• try and try!, send

• responds\_to?

• dup, deep\_dup, deep\_clone

• delegates (not mentioned in the book)

Date: 26-Jun-2016

**QUICK QUIZ QUESTIONS**

1. %s(foo) → ?
2. %Q[Hello “World”] → ?
3. foo = “Hello”  
   %W!#{foo} world! → ?
4. %w!#{foo} world! → ?
5. nil.object\_id == false.object\_id → ?
6. '123' === String → ?  
   ‘123’.is\_a?(String)  
   '123'.object\_id == '123'.object\_id → ?
7. ===, == and is\_a?

* f, \*rest, l = ["a", "b", "c", "d"]

rest → ?

* def dubSplat(a, \*b, \*\*c)

dupSplat(1, 2, 3, 4, k: 1, b: 2)

a, b, c → ?

1. Duck Typing?
2. p vs puts

val1 = true and false

val2 = true && false

val1 → ?, val2 → ?

1. class A

attr\_accessor :b

end

a = A.new

b = a

b.b = 1

a.b → ?

1. super vs super()
2. try, try!, send and respond\_to?

a = { a: 1, b: { k: 1, c: 3}}

b = a.dup

b[:b][:k] = 2

a[:b][:k] → ?

b = a.deep\_dup

b[:b][:k] = 2

a[:b][:k] → ?

1. delegate?

**Chapter 4 Questions:**

* What is [] in ruby?
* (0..-1).to\_a
* a[0..-1]
* a = [ 1, 3, 5, 7, 9 ]
* a[-3, 4]
* a[1,1]=[9,8, 7]
* unshift and shift
* a.first(2) and a.first
* scan
* has\_key?(:k), hash[:k]
* sort\_by?
* each, map/collect
* ways to pass block
* [1, 2, 3].each.index., each\_with\_index

The with\_index method takes an optional parameter to offset the starting index. each\_with\_indexdoes the same thing, but has no optional starting index.

For example:

[:foo, :bar, :baz].each.with\_index(2) do |value, index|

puts "#{index}: #{value}"

end

[:foo, :bar, :baz].each\_with\_index do |value, index|

puts "#{index}: #{value}"

end

* inject, reduce
* monkey patch
* enums are objects(external iterators) that just turn iterators without block into enum and enums then can be chained.
* enumerator(class) enumerable(module)
* blocks vs proc vs lambdas (time)

**Chapter 5 Question:**

* BasicObject.to\_s
* private constructor in ruby?

private :new

* memoization
* + vs concat for strings
* include vs extend a module
* multiple inheritance in ruby?