RUBY NOTES:

1. puts “hello”

Result: hello

=> nil

Nil is printed on next line because of ‘s’ in puts

If you write `put “hello”`, it will give result:

Hello => nil

Nil here is the return value

The print command just takes whatever you give it and prints it to the screen. puts (for "put string") is slightly different: it adds a new (blank) line after the thing you want it to print. You use them like this:

puts "What's up?" print "Oxnard Montalvo"

No parentheses or semicolons needed!

1. Ruby returns last code line in return

Ex: def greet

“Hello there”

end

Result: =>hello there

We have not written `return` here but still ‘hello there` is written because it is written before `end ` (or is a last code line to be returned)

1. Few ruby IRB examples:

Ex 1: $ def greet

$  return "Hello there"

$ end

=> nil

Ex2: $ def greet2

$  "no return necessary" (this is a text, ex: “hi”)

$ end

=> nil

Ex1.1 :$ greet (calling above defined function ‘greet’)

=> "Hello there"

Ex 2.1: $ greet2 (calling above defined function ‘greet2’)

=> "no return necessary"

3. A **method** (or message) in object-oriented **programming** (OOP) is a procedure associated with an object class. An object is made up of behavior and data. Data is represented as properties of the object and behavior as **methods**. A method is a programmed procedure that is defined as part of a [class](http://whatis.techtarget.com/definition/class) and included in any [object](http://searchsoa.techtarget.com/definition/object) of that class. A class (and thus an object) can have more than one method. A method in an object can only have access to the data known to that object, which ensures data integrity among the set of objects in an application. A method can be re-used in multiple objects.

1. To reverse the string, user method “reverse”

Eg: “<string>”.reverse

Input -> “test”.reverse

Output -> tset

1. You can always convert between different types using Ruby's "to" methods.

**to\_s** converts values to **s**trings.

**to\_i** converts values to **i**ntegers (numbers.)

**to\_a** converts values to **a**rrays.

1. To find the maxium number

Use [list of number].max

Eg: [10,20,30,40,90].max

Output: 90

1. To find min number

Use [list of numbers].min

Eg: [10,20,30,90].min

Output: 10

8. ticket.sort!

You might notice that the method has an exclamation point at its end. This just signals that we intend for Ruby to directly modify the same array that we've built, rather than make a brand new copy that is sorted. You'll notice that if you try calling ticket again, it will be sorted permanently!

If ticket has value [12,47,35] and we do ticket.sort!, the result is: [12,35,47]

9. Exponentiation raises one number (the base) to the power of the other (the exponent). For example, 2\*\*3 is 8, since 2\*\*3means "give me 2 \* 2 \* 2" (2 multiplied together 3 times).

--Modulo returns the remainder of division. For example, 25 % 7 would be 4, since 7 goes into 25 3 times with 4 left over.

10. Call .upcase on your name to make your name ALL CAPS and use putsto print it to the screen, like this:

puts "eric".upcase *# ==> ERIC*

On the next line, call .downcase to make your name all lower case. Make sure to use puts so you can see it printed out!

11. irb(main):010:0> **def** **hi**

irb(main):011:1> puts "Hello World!"

irb(main):012:1> **end** => :hi

Ruby’s response => :hi tells us that it knows we’re done defining the method.

Example: irb(main):015:0> **def** **hi**(name)

irb(main):016:1> puts "Hello #{name}!"

irb(main):017:1> **end**

* :hi
* irb(main):018:0> hi("Matz")
* Hello Matz!
* => nil

**Creating arrays**

Ruby arrays can be created with either literal notation or the Array.newconstructor.

**Syntax -1**

# Array.new constructor

variable = Array.new([repeat], [item])

**Example**

empty\_arr = Array.new => []

where empty\_arr is a variable

matzes = Array.new(3, "Matz!") => ["Matz!", "Matz!", "Matz!"]

where matzes is a variable

**Syntax -2**

# Array.new copy constructor

variable = Array.new(some\_array)

**Example**

more\_matzes = Array.new(matzes)

=> ["Matz!", "Matz!", "Matz!"]

**Syntax -3**

# Array literal notation

variable = []

other\_variable = [item1, item2, ..., itemN]

**Example**

bobbies = ["Bobby!", "Bobby!", "Backend Bobby!"]

=> ["Bobby!", "Bobby!", "Backend Bobby!"]